UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
Matematika - uporabna smer

ROBERT PETEK

RSA KRIPTOSISTEM

Diplomsko delo

Ljubljana, December 2000

KAZALO

PROGRAM DIPLOMSKEGA DELA

RSA KRIPTOSISTEM

Delo naj predstavi matematične osnove potrebne za razumevanje in implementacijo kriptosistema z javnimi ključi RSA. Eden od glavnih ciljev naj bo učinkovita implementacija digitalnega podpisa, generiranja javnih in privatnih ključev in šifriranje z RSA kriptosistemom ter analiza najbolj znanih napadov na ta sistem.

Literatura:

A. MENEZES, P. van OORSCHOT, S. VANSTONE: *Handbook of Applied Cryptography*, 4th ed., CRC Press 1999.

IEEE P1363 Standard Specifications for Public Key Cryptography, 1999, (glej http://grouper.ieee.org/groups/1363/index.html).

D. BONEH: *Twenty Years of Attacks on the RSA Cryptosystem*, Notices of the AMS, Vol. **46**/2, 1999, 203-213.

POVZETEK

RSA kriptosistem spada med kriptositeme z javnimi ključi. Uporablja se za zagotavljanje zasebnosti in pristnosti podatkov. Varnost RSA kriptosistema temelji na težavnosti problema RSA in problema faktorizacije velikih števil. Diplomsko delo obsega opis, analizo in implementacijo RSA kriptosistema. Uvodni poglavji predstavita kriptografijo in matematična orodja, potrebna za razumevanje RSA kriptosistema. Tretje poglavje opiše RSA kriptosistem in njegovo uporabo v RSA šifrirni shemi in shemi RSA elektronskega podpisa. Sledi obravnava napadov na RSA kriptosistem in s tem povezane varnosti. Kot študent uporabne matematike sem poudarek posvetil implementaciji RSA kriptosistema. Poglavje o implementaciji vsebuje pregled algoritmov, ki sem jih uporabil pri programu "RSA Encrypter&Signer v1.0". Program je priložen diplomskemu delu na CD-ROM-u. Omogoča generiranje RSA ključev, šifriranje po RSA šifrirni shemi in elektronski podpis po shemi IFSSA, ki je del standarda IEEE P1363. Namen diplome je predstaviti bralcu pomembnost javne kriptografije, konkretno RSA kriptosistema, in mu z zgledom olajšati morebitno implementacijo tega ali kakega podobnega kriptosistema.

Ključne besede: RSA kriptosistem, napadi na RSA, implementacija RSA, kriptografija, šifriranje, elektronski podpis, Evklidov algoritem, kongruence, primitivni elementi, praštevila, faktorizacija, praštevilskost

Matematična predmetna klasifikacija (2000): 94A60, 68P25, 11A05, 11A07, 11A41, 11A51

ABSTRACT

RSA cryptosystem is a public-key cryptosystem. It provides privacy and ensures authenticity of digital data. Security of RSA cryptosystem is based on intractability of RSA problem and integer factorization problem. Diploma consists of description, analysis and implementation of RSA cryptosystem. First two chapters present cryptography and mathematical tools for understanding RSA cryptosystem. Third chapter describes RSA cryptosystem and it's use in RSA encryption scheme and RSA signature scheme. Chapter 4 focuses on attacks on RSA cryptosystem and related security. As a student of applied mathematics I emphasized the implementation of RSA cryptosystem. The chapter about implementation includes descriptions of algorithms, which were used to develop the enclosed software "RSA Encrypter&Signer v1.0". It supports generation of RSA keys, encryption with RSA encryption scheme and digital signature with IFSSA scheme, included in the IEEE P1363 standard. My main goal was to introduce the reader to public-key cryptography, in particular to RSA cryptosystem, and with my implementation aid eventual developers of this or any similar cryptosystem.

Key words: RSA cryptosystem, attacks on RSA, implementation of RSA, cryptography, encryption, digital signatures, Euclidean algorithm, congruences, primitive roots, primes, factorization, primality

Mathematics Subject Classification (2000): 94A60, 68P25, 11A05, 11A07, 11A41, 11A51

1. UVOD

Želja po zasebnosti je bila vedno pomemben del človekovega značaja. Tako ostaja tudi danes in se z uvajanjem informacijskih tehnologij še posebej izpostavlja. Internet v osnovi ni bil zgrajen s podporo zasebnosti. Pretok informacij je potekal nezavarovano in vsakdo z zadostnim znanjem in priložnostjo je lahko prišel do njih. Za vedno večje število uporabnikov, ki v tem navideznem svetu preživimo vedno več časa, je rešitev problema zasebnosti šifriranje. Šifriranje spada v področje kriptografije, vede, ki se v splošnem ukvarja z načini za dosego informacijske varnosti. Šifriranje je le eden od kriptografskih temeljev. V času, ko se velik del trgovanja seli na internet, se je med uporabniki elektronskega poslovanja pojavila tudi potreba po varnosti in pristnosti podatkov ter pristnosti oseb. Kriptografski temelj, ki služi ugotavljanju pristnosti, je elektronski podpis. Podobne potrebe lahko opazimo tudi v trenutno zelo zanimivih mobilnih tehnologijah, kot so prenosni računalniki in mobilna telefonija.

Kot študent uporabne matematike sem želel pridobljeno znanje uporabiti na način, ki bi povezal svet matematike s svetom, v katerem živimo. Kriptografija postaja pomemben del informacijskih tehnologij in jo v vsakdanjem življenju vedno bolj uporabljamo. Mentor, Dr. Aleksandar Jurišić, ki se mu za vso pomoč tudi iskreno zahvaljujem, mi je v tej želji za temo diplomske naloge predlagal RSA kriptosistem, verjetno eden od najbolj uporabljenih kriptosistemov ta čas, ki je hkrati tudi zelo priročen za implementacijo. Tema diplomskega dela obsega opis in analizo RSA kriptosistema ter implementacijo šifriranja in elektronski podpis po standardu IEEE P1363 [14].

RSA kriptosistem, katerega avtorji so R.L. Rivest, A. Shamir in L. Adleman [21], spada med kriptositeme z javnimi ključi. RSA je prva realizacija javne kriptografije, o kateri sta prva spregovorila W. Diffie in M.E. Hellman leta 1976 [11]. Uporablja se za zagotavljanje zasebnosti in tudi pristnosti podatkov. Njegove implementacije se pojavljajo tako v programski kot tudi v strojni opremi. Varnost RSA kriptosistema temelji na težavnosti problema RSA in problema faktorizacije velikih števil. Analiza varnosti RSA kriptosistema je dala nekatere zanimive napade nanj, vendar nobeden od njih ni bil usoden. Največkrat ti napadi opozarjajo na nepravilno uporabo RSA kriptosistema. Čeprav je RSA kriptosistem relativno enostaven v primerjavi z drugimi javnimi kriptosistemi, varna implementacija le-tega nikakor ni trivialna naloga.

Organizacija diplomskega dela je sledeča. V preostanku poglavja bomo na kratko spregovorili o kriptografiji in podali najosnovnejše definicije, osnovno terminologijo in formalne modele naslednjih kriptografskih temeljev: kriptosistem, zgoščevalne funkcije in elektronski podpis. V drugem poglavju bomo podali osnovna matematična orodja: osnovne definicije in dejstva v kolobarju celih števil in ostankov po modulu n. Oboje bomo potrebovali pri razumevanju RSA kriptosistema in obravnavanju napadov nanj. Na koncu drugega poglavja se bomo posvetili teoriji zahtevnosti. Ta nam bo v pomoč pri analizi zahtevnosti implementacije. V tretjem poglavju bomo opisali RSA kriptosistem, problem RSA in problem faktorizacije ter opisali RSA šifrirno shemo in shemo RSA elektronskega podpisa. Sledi poglavje o napadih na RSA kriptosistem. Z napadi ugotavljamo, kako varen je RSA. Obravnavali bomo pet razredov napadov. Za varnost RSA so najpomembnejši napadi povezani s problemom faktorizacije. V petem poglavju bomo pripravili vse, kar je potrebno za implementacijo RSA kriptosistema. Opisali bomo tudi postopek za ugotavljanje praštevilskosti in iskanje praštevil, potrebnih za generiranje RSA ključev. V zadnjem poglavju bomo predstavili probleme, s katerimi se srečujemo pri implementaciji RSA v praksi in opisali možnosti uporabe RSA. V zaključku poglavja bomo povedali še nekaj o patentih in standardih, povezanih z RSA kriptosistemom. Poglavje o implementaciji se neposredno nanaša na program *RSA Encrypter&Signer v1.0*, ki je bil napisan za to diplomsko delo in je priložen na CD-ROM-u. Program vsebuje generiranje RSA ključev, šifriranje po RSA šifrirni shemi in elektronski podpis po shemi IFSSA novonastalega standarda IEEE P1363, ki uporablja zgoščevalno funkcijo SHA-1.

1.1 Kriptografija

Kriptografija ima dolgo in zanimivo zgodovino. Njeni začetki segajo v čas Egipčanov, 4000 let nazaj. V novejši zgodovini so kriptografijo uporabljale v glavnem vojaške, diplomatske in vladne službe nasplošno. Iznajdba radia je dala kriptografiji izreden polet. Med drugo svetovno vojno je na dogodke močno vplivala uporaba, zloraba in razbijanje kriptografskih sistemov, ki so temeljili na radijskih valovih. Več o zgodovini kriptografije lahko izvemo v Kahnovi knjigi *The Codebreakers* [16]. S širitvijo in uveljavitvijo uporabe računalnikov in računalniških sistemov se je v šestdesetih letih tudi v privatnem sektorju pojavila potreba po sredstvih za zaščito podatkov v digitalni obliki in po varnostnih storitvah. Kriptografija dobiva nov zagon z razcvetom interneta, elektronskega poslovanja in mobilnimi tehnologijami, ki je potrebo po teh storitvah še izdatno povečalo.

Zapišimo najprej formalno definicijo kriptografije in z njo povezanih pojmov.

DEFINICIJA *Kriptografija* je veda o matematičnih sredstvih za dosego informacijske varnosti, kot je zasebnost, zaupnost, celovitost podatkov, pristnost oseb in/ali podatkov.

DEFINICIJA *Kriptoanaliza* je študij o matematičnih tehnikah za premagovanje kriptografskih sredstev in, bolj splošno, informacijsko varnostnih storitev.

DEFINICIJA *Kriptologija* je študij kriptografije in kriptoanalize.

Z drugimi besedami, kriptografija se nanaša na izdelovanje kriptografskih sistemov, medtem ko se kriptoanaliza ukvarja z razbijanjem le-teh. Kriptografija je nastala kot posledica komunikacije v prisotnosti nasprotnikov oziroma tekmecev. Uporablja za preprečevanje in odkrivanje zlorab in ostalih zlonamernih dejanj. Eden od ciljev kriptografije je npr. zasebnost - dve osebi se želita pogovarjati tako, da nasprotnik ne ve nič o tem, kaj se pogovarjata. Z razvojem kriptografije se je število ciljev povečevalo, prav tako pa tudi število orodij, s katerimi lahko te cilje dosežemo. Kriptografija nudi metode, ki omogočajo naslednje cilje:

- *Zasebnost*. Nasprotnik ne izve nič koristnega iz poslanega sporočila.
- *Zaupnost*. Vsebina podatkov je dostopna le tistim, ki so za dostop pooblaščeni.
- *Pristnost*. Sprejemnik sporočila se lahko sam prepriča oziroma preveri ali je sporočilo res poslal naveden pošiljatelj.
- *Celovitost*. Sprejemnik ima zagotovilo, da prispelo sporočilo ni bilo spremenjeno (npr. s strani nasprotnika).
- *Podpis*. Sprejemnik sporočila lahko prepriča tretjo osebo, da je sprejeto sporočilo celovito in izvira od navedenega pošiljatelja.
- *Preprečevanje nepriznavanja*. To pomeni preprečiti osebi, da bi zanikala predhodno obveznost ali dejanje.
- *Istočasna menjava*. Vredno sporočilo, kot je podpis ali pogodba, ni poslano, dokler ni sprejeto neko drugo vredno sporočilo, npr. podpis druge osebe.
- *Uskladitev*. Pri pogovoru več oseb, so le-te zmožne uskladiti svoje aktivnosti proti skupnemu cilju, četudi v prisotnosti nasprotnika.

Osnovni cilj kriptografije je zadostna podpora tem področjem tako v teoriji kot praksi. V zadnjih dvajsetih letih se je kriptografija razvila v eksaktno vedo. Trenutno obstaja več mednarodnih znastvenih konferenc (CRYPTO, ASIA-CRYPT, EUROCRYPT) in znanstvena organizacija International Association for Cryptologic Research (IACR), ki se ukvarjajo izključno z raziskovanjem na področju kriptografije.

1.2 Osnovna terminologija

Preden spoznamo osnovne kriptografske temelje: kriptosistem, zgoščevalne funkcije in elektronski podpis, bomo opisali osnovne termine, ki jih srečamo pri njihovi obravnavi.

- A označuje končno množico imenovano *osnovna abeceda*. Npr. $A = \{0, 1\}$, binarna abeceda, je pogosto uporabljena osnovna abeceda. Vsaka druga abeceda je lahko opisana z binarno abecedo.

- M označuje *prostor sporočil*. M vsebuje nize sestavljene iz znakov osnovne abecede. Elemente M imenujemo *sporočilo* ali *čistopis*. Npr. M lahko vsebuje binarne nize, navadne besede, računalniško kodo, itd.

- C označuje *prostor šifriranih sporočil*. C vsebuje nize sestavljene iz znakov osnovne abecede. Elemente C imenujemo *šifrirano sporočilo* ali *tajnopis*.

- K označuje *prostor ključev*. Element množice K imenujemo *ključ*.

- Vsakemu ključu $e \in K$ priredimo bijekcijo E_e iz prostora sporočil M v prostor šifriranih sporočil C. Funkcijo E_e imenujemo *šifrirna funkcija* (ali *transformacija*). Proces uporabe šifrirne funkcije E_e na sporočilu $m \in M$ imenujemo *šifriranje* ali *enkripcija*.

- Vsakemu ključu $d \in K$ priredimo bijekcijo D_d iz prostora šifriranih sporočil C v prostor sporočil M. Funkcijo D_d imenujemo *dešifrirna funkcija* (ali *transformacija*). Proces uporabe dešifrirne funkcije D_d na sporočilu $m \in M$ imenujemo *dešifriranje* ali *dekripcija*.

- S označuje *prostor podpisov*. Pogost primer so npr. binarni nizi fiksnih dolžin.

- S_A je funkcija iz prostora sporočil M v prostor podpisov S in jo imenujemo *podpisna funkcija* (ali *transformacija*) osebe A. Transformacijo S_A varuje oseba A in jo uporablja za *podpisovanje* sporočil iz M.

- V_A je funkcija iz prostora $M \times S$ v prostor $\{Da, Ne\}$. V_A imenujemo *funkcija preverjanja podpisov* osebe A, je javno znana in jo uporabljajo druge osebe za *preverjanje podpisov*, ki jih je naredila oseba A.

1.3 Kriptosistemi

Kriptosistem je v splošnem pojem, ki se nanaša na niz kriptografskih temeljev za doseganje kriptografskih ciljev v informacijsko-varnostnih storitvah. Največkrat je pojem vezan na temelje v povezavi z zasebnostjo, v splošnem, na šifriranje. Konkretno implementacijo nekega kriptosistema za šifriranje imenujemo *šifrirna shema*. Oglejmo si definicijo kriptosistema.

DEFINICIJA *Kriptosistem* je peterica (M, C, K, E, D), za katero velja:
1. M je končna množica možnih sporočil,
2. C je končna množica možnih šifriranih sporočil,
3. K je končna množica ključev,
4. za vsak ključ $k \in K$ imamo šifrirno transformacijo $E_k \in E$ in ustrezeno dešifrirno transformacijo $D_k \in D$. $E_k : M \to C$ in $D_k : C \to M$ sta taki funkciji, da je $D_k(E_k(x)) = x$ za vsak $x \in M$.

Kriptosistemi se delijo na dva osnovna tipa:

(i) Simetrični kriptosistem. Standardna kriptografska rešitev za problem zasebnosti je kriptosistem s simetričnim ključem. Kjuč je en sam in skrbno varovana skrivnost. Govorimo o *tajnem* (ali *skrivnem*) *kjuču*. Šifrirna in dešifrirna transformacija E_k in D_k pri danem kjuču $k \in K$ sta bodisi enaki bodisi se iz ene transformacije da na enostaven način dobiti drugo. Primer simetričnega kriptosistema je DES (Data Encryption Standard) [23, str. 70].

(ii) Javni kriptosistem. Javni kriptosistem je kriptositem, kjer je del ključa javno znan, del pa je zaseben. Ideja je ta, da je v praksi računsko neizvedljivo poiskati ustrezno dešifrirno transformacijo D_k iz dane šifrirne transformacije E_k. Če je to tako, potem lahko objavimo šifrirno transformacijo E_e, kjer je e javni del ključa k (ali *javni ključ*). Prednost javnega kriptosistema pred simetričnim je v tem, da lahko pošiljamo šifrirana sporočila, ne da bi se poprej zmenili za skrivni ključ. Oseba, kateri pošljemo sporočilo, šifrirano z njenim javnim ključem e, je po zgornjem edina, ki lahko to sporočilo dešifrira z *zasebnim* (ali *privatnim*) *ključem* d. Primer javnega kriptosistema, kateremu se bomo tudi posvetili, je RSA kriptosistem (§3.1).

Zakaj so ključi sploh potrebni? Če imamo transformacije, parametrizirane s ključi, potem nam v primeru, da je določena šifrirna/dešifrirna transformacija odkrita, ne bo potrebno zgraditi nove šifrirne sheme ali sheme elektronskega podpisa, pač pa bomo zamenjali le kjuč. V praksi je dobro pogosto menjavati ključe (saj se s tem spremenijo tudi šifrirne/dešifrirne transformacije).

Osnovna lastnost v kriptografiji je, da so množice M, C, K, $E = \{E_e \mid e \in K\}$ in $D = \{D_d \mid d \in K\}$ javno znane. Varno komuniciranje dveh oseb z izbranim kriptosistemom predpostavlja varovanje uporabljenega ključa, ki ga morata ti dve osebi tudi izbrati. Dodatno varnost lahko pridobimo z varovanjem nekega razreda šifrirnih/dešifrirnih transformacij, vendar graditi varnost celotnega sistema na tej osnovi ni priporočljiva. Zgodovina je že pokazala, da je vzdrževanje varnosti šifrirnih/dešifrirnih transformacij zelo težka naloga.

DEFINICIJA Pravimo, da je kriptosistem *zlomljiv*, če lahko tretja oseba, brez prejšnjega poznavanja ključa, sistematično pridobi sporočilo iz šifriranega sporočila v nekem primerno določenem časovnem okvirju.

Primerno določen časovni okvir je funkcija uporabnega časa, v katerem morajo biti podatki zavarovani. Npr. naročilo za prodajo neke delnice je lahko varovana skrivnost le nekaj minut, državna skrivnost pa se lahko varuje tudi več desetletij.

1.4 Zgoščevalne funkcije

Eden od pomembnih temeljev moderne kriptografije predstavljajo zgoščevalne funkcije (angl. hash function), imenovane tudi enosmerne zgoščevalne funkcije. Zgoščevalne funkcije se uporabljajo za preverjanje celovitosti podatkov v povezavi z elektronskim podpisom, kjer na sporočilu najprej uporabimo zgoščevalno funkcijo, potem pa zgoščeno vrednost (angl. hash-value), kot predstavnika tega sporočila, podpišemo. Formalna definicija zgoščevalne funkcije je naslednja.

DEFINICIJA *Zgoščevalna funkcija* je v splošnem funkcija h, ki preslika binaren vnosni niz x poljubne dolžine v binaren niz $h(x)$ fiksne dolžine. Vrednost $h(x)$ imenujemo *zgoščena vrednost*.

Zgoščevalna funkcija mora imeti vsaj dve lastnosti: *stiskanje* (veliko definicijsko območje slika v majhno zalogo vrednosti) in *enostavnost računanja*. Osnovna ideja zgoščevalnih funkcij je ta, da zgoščena vrednost predstavlja kompaktno reprezentativno sliko vnosnega niza, imenovano tudi *digitalni odtis*, ki jo lahko uporabimo tako, kot da bi bila enolično določena z vnosnim nizom.

Zgoščevalne funkcije iz sporočila izdelajo zgoščeno vrednost. Bolj natančno, zgoščevalna funkcija h preslika niz bitov poljubne dolžine v niz določene dolžine, npr. n bitov. Funkcija h z definicijskim območjem D in zalogo vrednosti R, kjer $|D| > |R|$, slika več elementov v eno sliko, zato prihaja do t.i. *trkov*. Če funkcijo h omejimo na definicijsko območje z elementi dožine t, $t > n$, in je h 'naključna' v smislu, da je vsaka slika enako verjetna, potem ima približno 2^{t-n} elementov isto sliko, dva naključno izbrana elementa pa z verjetnostjo 2^{-n} isto sliko (neodvisno od t).

Tipičen primer uporabe zgoščevalnih funkcij je naslednji. Ob trenutku T_1 izračunamo zgoščeno vrednost sporočila x. Pristnost te vrednosti (ne pa tudi sporočila) na nek način ohranimo. Čez čas, v trenutku T_2, opravimo naslednji test, da ugotovimo, ali je bilo sporočilo spremenjeno. Izračunamo zgoščeno vrednost sporočila x' in jo primerjamo z varovano zgoščeno vrednostjo x. Če sta vrednosti enaki, potem verjamemo, da sta tudi sporočili enaki oziroma da sporočilo ni bilo spremenjeno. Problem ohranjanja velikega sporočila se zmanjša na problem ohranjanja zgoščene vrednosti. Ker za zgoščevalne funkcije obstaja verjetnost trkov, mora biti zgoščena vrednost enolično določena s sporočilom vsaj v praksi. Iskanje trkov mora biti računsko neizvedljivo (v praksi se trki takorekoč ne smejo pojavljati). Primeri zgoščevalnih funkcij so MD4 [23, str. 247], RIPMED-160 [19, str. 349] in SHA-1 (Secure Hash Algorithm - revised) (§5.8).

1.5 Elektronski podpis

Elektronski podpis je kriptografski temelj za doseganje pristnosti in dodeljevanje pooblastil ter za preprečevanje nepriznavanja storjenih dejanj in obveznosti. Namen elektronskega podpisa je priskrbeti način povezovanja identitete neke osebe z informacijo. Proces podpisovanja iz sporočila in zasebne informacije osebe naredi podpis. Formalna definicija elektronskega podpisa je naslednja.

DEFINICIJA *Elektronski podpis* je podatkovni niz v digitalni obliki namenjen zagotavljanju pristnosti in/ali porekla podatkov.

Obstajata dva splošna razreda shem elektronskih podpisov:

(i) Shema elektronskega podpisa s sporočilom ne zahteva originalno sporočilo pri procesu preverjanja. V tem primeru se originalno sporočilo pridobi iz podpisa. Primer take sheme je shema RSA elektronskega podpisa (§3.4).

(ii) Shema elektronskega podpisa z dodatkom zahteva originalno sporočilo pri procesu preverjanja. Primera shem elektronskega podpisa z dodatkom sta DSS (Digital Signature Standard) [23, str. 209] in IFSSA (Integer Factorization Signature Scheme with Appendix) (§5.9).

Shema elektronska podpisa je sestavljena iz dveh delov: *procesa podpisovanja* in *procesa preverjanja*. Oglejmo si primer sheme elektronskega podpisa z dodatkom.

Proces podpisovanja. Oseba A (*podpisnik*) naredi podpis s za sporočilo $m \in M$ takole:
1. Izračuna $s = S_A(m)$, kjer je S_A podpisna funkcija.
2. Pošlje par (m, s).

Proces preverjanja. Preverjanje podpisa osebe A opravi oseba B takole:
1. Pridobi funkcijo preverjanja V_A od osebe A.
2. Izračuna $u = V_A(m, s)$.
3. Sprejme podpis osebe A, če je $u = Da$, zavrne podpis, če je $u = Ne$.

Transformaciji S_A in V_A sta ponavadi določeni s ključem, tj. razred transformacij podpisovanja in preverjanja je javno znan, vsaka transformacija pa je določena s ključem. Transformacija podpisovanja S_A osebe A je določena s ključem k_A, ki jo mora le-ta varovati. Podobno je transformacija preverjanja V_A osebe A določena s ključem l_A, ki pa je javno znan.

Shemo elektronskega podpisa lahko dobimo tudi iz javne šifrirne sheme. Naj bo E_e šifrirna transformacija na prostoru sporočil M v šifrirni prostor C. Predpostavimo, da je $M = C$. Naj bo D_d ustrezna dešifrirna transformacija. Potem velja

$$D_d(E_e(m)) = E_e(D_d(m)) = m$$

za vse $m \in M$. Taki javni šifrirni shemi pravimo *obrnljiva šifrirna shema*. Predpostavka $M = C$ je potrebna, da velja enakost za vse $m \in M$, sicer $D_d(m)$ ni definiran za $m \notin C$.

Konstrukcija sheme elektronskega podpisa iz obrnljive javne šifrirne sheme poteka takole:
1. Naj bo M prostor sporočil za šifrirno shemo in $C = M$ prostor podpisov S.
2. Naj bosta (e, d) par ključev za javno šifrirno shemo.
3. Podpisno funkcijo S_A definiramo kot D_d, tj. podpis za sporočilo $m \in M$ je enako $s = D_d(m)$.
4. Funkcijo preverjanja V_A definiramo takole:

$$V_A(m, s) = \begin{cases} Da, & \text{če } E_e(s) = m \\ Ne, & \text{sicer.} \end{cases}$$

V tem primeru govorimo seveda o shemi elektronskega podpisa z dodatkom. Podobno lahko naredimo v primeru sheme elektronskega podpisa s sporočilom. Ker sporočilo dobimo iz podpisa ($m = E_e(s)$), vzamemo za funkcijo preverjanja

$$V_A(s) = \begin{cases} Da, & \text{če } E_e(s) \in M' \\ Ne, & \text{sicer,} \end{cases}$$

kjer je $M' \subseteq M$ prostor sporočil, katerega elementi so izbrane oblike. Npr. sporočilo se začne z določenim znakom ali številom. Primer obrnljive šifrirne sheme je RSA šifrirna shema (§3.3).

Elektronski podpis je v praksi uporaben, če ga lahko podpisnik enostavno izračuna, prejemnik enostavno preveri in je varen pred ponaredbo v času njegove veljavnosti. Podpis $s \in S$ za sporočilo $m \in M$ je veljaven natanko takrat, ko $V_A(m, s) = Da$. Varnost pred ponaredbo pa pomeni, da je za vse, z izjemo podpisnika, računsko neizvedljivo poiskati tak $m \in M$ za $s \in S$, da je $V_A(m, s) = Da$ (vsaj v času veljavnosti podpisa).

2. MATEMATIČNA ORODJA

V tem poglavju bomo opisali matematična orodja, ki jih bomo potrebovali v nadaljnih poglavjih. RSA kriptosistem uporablja števila ostankov po modulu zelo velikega števila. Modul je produkt dveh različnih praštevil. Zato nas bodo v prvih dveh razdelkih zanimale osnovne definicije in dejstva povezana s celimi števili, praštevili in števili ostankov po modulu *n*, skratka teorija števil (viri [1], [6], [12] in [24]). Ta nam bo podlaga za razumevanje RSA kriptosistema in napadov nanj. Poglavje bomo zaključili s pregledom teorije zahtevnosti (vira [19] in [20]). Potrebovali jo bomo pri analizi računskih problemov in ocenjevanju zahtevnosti algoritmov, ki te probleme rešujejo.

2.1 Cela števila

Pri kolobarju celih števil **Z** bomo ponovili osnovne definicije iz teorije števil. Ena od osnovnih lastnosti v **Z** je ta, da za dani celi števili a in b, kjer $a \geq 0$ in $b > 0$, vedno obstajata enolični celi števili q in r, tako da velja $a = qb + r$, kjer $q \geq 0$ in $0 \leq r < b$ (*izrek o deljenju* [6, str. 3]). Ker za praštevila ni enostavne formule, za neko veliko število ni enostavno ugotoviti, ali je praštevilo, ali ne. Zato bomo v tem razdelku naredili pregled uporabnih znanih dejstev o praštevilih.

DEFINICIJA Naj bosta a in b celi števili. Pravimo, da število a *deli* število b (ekvivalentno, a je *delitelj* ali *faktor* za b), če obstaja tako število c, da je $b = ac$. Pišemo $a \mid b$.

DEFINICIJA Deljenje celih števil a in b, kjer $b \geq 1$, nam natanko določi števili q in r, tako da je $a = qb + r$, kjer $0 \leq r < b$. Število q imenujemo *kvocient* in ga označimo z a div b. Število r imenujemo *ostanek*, pišemo a mod b.

DEFINICIJA Število d je *največji skupni delitelj* števil a in b, pišemo $d = D(a, b)$, če $d \mid a$ in $d \mid b$ in za vsako število c, ki deli a in b, velja $c \mid d$. Definiramo $D(a, 0) = a$.

DEFINICIJA Število d je *najmanjši skupni večkratnik* števil a in b, pišemo $d = v(a, b)$, če $a \mid d$ in $b \mid d$ in za vsako število c, ki ga delita a in b, velja $d \mid c$.

DEFINICIJA Pravimo, da sta si števili a in b *tuji*, če je njun največji skupni delitelj enak 1.

DEFINICIJA Število $p \geq 2$ imenujemo *praštevilo*, če sta 1 in p edina pozitivna delitelja za p. Vsa druga števila imenujemo *sestavljena* števila.

Vsako število $n \geq 2$ ima faktorizacijo, ki je produkt potenc praštevil: $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, kjer so p_i različna praštevila in $e_i \geq 0$. To nam pove *osnovni izrek o aritmetiki* [24, str. 142]. Še več, faktorizacija je enolično določena do vrstnega reda potenc natančno.

Največji skupni delitelj števil $a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ in $b = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$, kjer $e_i \geq 0$ in $f_i \geq 0$, dobimo tako, da zmnožimo potence skupnih praštevil p_i z manjšim od eksponentov e_i in f_i. Torej

$$D(a, b) = p_1^{\min\{e_1, f_1\}} p_2^{\min\{e_2, f_2\}} \cdots p_k^{\min\{e_k, f_k\}}.$$

Podobno izračunamo najmanjši skupni večkratnik števil a in b takole

$$v(a, b) = p_1^{\max\{e_1, f_1\}} p_2^{\max\{e_2, f_2\}} \cdots p_k^{\max\{e_k, f_k\}}.$$

Ni se težko prepričati, da velja enakost $v(a, b) \cdot D(a, b) = ab$.

Največji skupni delitelj števil a in b pa lahko izračunamo tudi brez poznavanja faktorizacije števil a in b in sicer z Evklidovim algoritmom (glej §5.3, algoritem 12). Evklidov algoritem pri danih celih številih a in b, $a > b > 0$, na vsakem koraku izračuna število $a_{i+2} = a_i \bmod a_{i+1}$, kjer je $a_0 = a$ in $a_1 = b$, in vrne zadnje od nič različno število zaporedja $\{a_i\}$. Evklidov algoritem temelji na preprostem dejstvu.

TRDITEV 2.1 Naj bosta a in b pozitivni celi števili, $a > b$. Potem velja: $D(a, b) = D(b, a \bmod b)$.

DOKAZ Naj bo $d = D(a, b)$. Potem $d \mid a$ in $d \mid b$. Ker je $a \bmod b = a - b \cdot a \operatorname{div} b$, velja tudi $d \mid a \bmod b$. Naj bo c poljubno število, ki deli b in $a \bmod b$. Potem c deli $b \cdot a \operatorname{div} b + a \bmod b = a$. Po definiciji največjega skupnega delitelja $c \mid d = D(a, b)$ in zato $d = D(b, a \bmod b)$. □

TRDITEV 2.2 Evklidov algoritem pri podatkih $a, b \in \mathbf{Z}$, $a > b$, vrne $D(a, b)$.

DOKAZ Zapišimo korake Evklidovega algoritma pri danih celih številih $a = a_0$ in $b = a_1$.

$$\begin{aligned}
a_0 &= q_0 a_1 + a_2, & 0 < a_2 < a_1 \\
a_1 &= q_1 a_2 + a_3, & 0 < a_3 < a_2 \\
&\dots \\
a_{k-2} &= q_{k-2} a_{k-1} + a_k, & 0 < a_k < a_{k-1} \\
a_{k-1} &= q_{k-1} a_k, & a_{k+1} = 0,
\end{aligned}$$

kjer je $q_i = a_i \operatorname{div} a_{i+1}$ in $a_{i+2} = a_i \bmod a_{i+1}$ za vsak i. Ker je $\{a_i\}$ strogo padajoče zaporedje pozitivnih celih števil, se postopek enkrat konča. Naj se konča pri $a_{k+1} = 0$. Trdimo, da je $a_k = D(a, b)$. Ker je $a_1 > a_2 = a_0 \bmod a_1$, po trditvi 2.1 velja $D(a_0, a_1) = D(a_1, a_2)$. Podobno je $a_2 > a_3 = a_1 \bmod a_2$ in $D(a_1, a_2) = D(a_2, a_3)$, itd. Potem je

$$D(a_0, a_1) = D(a_1, a_2) = \dots = D(a_k, 0) = a_k.$$

Evklidov algoritem je moč razširiti tako, da ne vrne le največji skupni delitelj d števil a in b, ampak tudi celi števili x in y, ki zadoščata Diofantski enačbi $ax + by = d$ (glej §5.3, algoritem 13). Razširjen Evklidov algoritem poleg števila $a_{i+2} = a_i \bmod a_{i+1}$ na vsakem koraku izračuna še števili $x_{i+2} = x_i - (a_i \operatorname{div} a_{i+1})x_{i+1}$, kjer je $x_0 = 1$, $x_1 = 0$, ter $y_{i+2} = y_i - (a_i \operatorname{div} a_{i+1})y_{i+1}$, kjer je $y_0 = 0$, $y_1 = 1$, in vrne poleg $D(a, b)$ še ustrezni števili zaporedja $\{x_i\}$ in $\{y_i\}$, rešitvi enačbe $ax + by = d$.

TRDITEV 2.3 Razširjen Evklidov algoritem pri podatkih $a, b \in \mathbf{Z}$, $a > b$, poišče taki števili $x, y \in \mathbf{Z}$, ki sta rešitvi enačbe $ax + by = d$, kjer $d = D(a, b)$.

DOKAZ Naj bo $a = a_0$ in $b = a_1$, kjer $a_0 > a_1 > 0$. Razširjen Evklidov algoritem nam po trditvi 2.2 da končno zaporedje $a_0, a_1, \dots, a_k$, za katerega velja $D(a_0, a_1) = D(a_1, a_2) = \dots = D(a_k, 0) = a_k = d$. Za zaporedji $\{x_i\}$ in $\{y_i\}$, kjer $x_0 = 1$, $x_1 = 0$, $x_{i+2} = x_i - q_i x_{i+1}$ in $y_0 = 0$, $y_1 = 1$, $y_{i+2} = y_i - q_i y_{i+1}$, trdimo, da velja $a_i = ax_i + by_i$ za vsak i. Za $x_0 = 1$, $y_0 = 0$ in $x_1 = 0$, $y_1 = 1$ velja

$$\begin{aligned}
a_0 &= a = ax_0 + by_0 \\
a_1 &= b = ax_1 + by_1.
\end{aligned}$$

Ker je $a_2 = a_0 \bmod a_1 = a_0 - q_0 a_1$, kjer $q_0 = a_0 \operatorname{div} a_1$, ga lahko izrazimo kot

$$\begin{aligned}
a_2 &= ax_0 + by_0 - q_0(ax_1 + by_1) \\
&= a(x_0 - q_0 x_1) + b(y_0 - q_0 y_1) \\
&= ax_2 + by_2
\end{aligned}$$

Podobno lahko $a_3 = a_1 \bmod a_2 = a_1 - q_1 a_2$ izrazimo kot $a_2 = ax_3 + by_3$, itd. Na koncu dobimo $a_k = ax_k + by_k$. Rešitvi, ki zadoščata enačbi $ax + by = d$ sta $x = x_k$ in $y = y_k$. □

TRDITEV 2.4 Naj bodo a, b in c cela števila. Potem ima enačba $ax + by = c$ rešitev v x, $y \in \mathbf{Z}$ natanko takrat, ko $D(a, b) \mid c$.

DOKAZ Naj obstajata taki števili x, $y \in \mathbf{Z}$, da velja $ax + by = c$ in naj bo $d = D(a, b)$. Potem velja $a_1dx + b_1dy = c$, kjer $a_1 = a/d$ in $b_1 = b/d$. Očitno $d = D(a, b) \mid c$. Za dokaz nasprotne smeri je dovolj pokazati, da ima enačba $ax + by = d$ rešitev. Naj ima enačba $ax + by = d$ rešitev v x_1 in y_1. Potem ima enačba $ax + by = c$ rešitev v $x = kx_1$ in $y = ky_1$, kjer $k = c/d$. Obstoj rešitve enačbe $ax + by = d$ nam da razširjen Evklidov algoritem (trditev 2.3). □

Ponovimo osnovna dejstva o praštevilih.[1] Edino sodo praštevilo je 2. Praštevilo $p > 2$ je gotovo liho, saj so vsa soda števila večkratniki števila 2. Če je praštevilo p deljivo s produktom števil a in b, potem velja $p \mid a$ ali $p \mid b$ (ali oboje). Na vprašanje, koliko je praštevil, je že pred dvema tisočletjema odgovoril Evklid.

TRDITEV 2.5 (Evklid) Praštevil je neskončno mnogo.

DOKAZ Naj bodo $p_1 = 2 < p_2 = 3 < \ldots < p_k$ vsa praštevila v $\mathbf{Z}$. Definirajmo $n = p_1p_2 \cdots p_k + 1$ in naj bo p praštevilo, ki deli n. Potem število p ne more biti nobeden od praštevil $p_1, p_2, \ldots, p_k$, sicer bi p delil razliko $n - p_1p_2 \cdots p_k = 1$. Torej je p novo praštevilo, kar je v protislovju s predpostavko. □

Zanimiva sta naslednja dva izreka o praštevilih, ki ju ne bomo dokazali. Prvi, *izrek o gostoti praštevil* [12, str. 9], nam da oceno, koliko je praštevil do nekega izbranega števila x, drugi izrek [1, str. 233] pa kako veliko je n-to praštevilo.

IZREK 2.6 (Izrek o gostoti praštevil) Naj bo $\pi(x)$ število praštevil $\leq x$. Potem je $\pi(x) \sim x/\ln x$.[2] □

IZREK 2.7 (Aproksimacija n-tega praštevila) Naj p_n označuje n-to praštevilo. Potem je $p_n \sim n \cdot \ln n$. Bolj natančno, za $n \geq 1$ velja $p_n > n \cdot \ln n$ in za $n \geq 6$ velja $n \ln n < p_n < n(\ln n + \ln\ln n)$. □

Izrek o gostoti praštevil pravi, da lahko za veliko število x aproksimiramo $\pi(x)$ z $x/\ln x$. Npr. za število $x = 10^{10}$ je $\pi(x) = 455{,}052{,}551$, medtem ko je $\lfloor x/\ln x \rfloor = 434{,}294{,}481$. Podobno si lahko z aproksimacijo pomagamo pri iskanju n-tega praštevila. Npr. za $n = 100000$ je praštevilo $p_n = 1299827$, medtem ko je $\lfloor n \ln n \rfloor = 1151292$ in $\lfloor n(\ln n + \ln\ln n) \rfloor = 1395639$.

DEFINICIJA Za celo število $n \geq 1$ naj $\phi(n)$ označuje število celih števil na intervalu $[1, n]$, ki so tuja številu n. Funkcijo ϕ imenujemo *Eulerjeva funkcija*.

TRDITEV 2.8 Lastnosti Eulerjeve funkcije:

(i) Če je p praštevilo, potem je $\phi(p) = p - 1$.

(ii) Če je $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ faktorizacija na praštevila, potem je

$$\phi(n) = \prod_{i=1}^{k} (p_i - 1) p_i^{e_i - 1} = n \cdot \prod_{i=1}^{k} (1 - \tfrac{1}{p_i}).$$

(iii) Eulerjeva funkcija je multiplikativna, tj. če velja $D(m, n) = 1$, potem je $\phi(mn) = \phi(m)\phi(n)$.

[1] Veliko zanimivosti o praštevilih na spletnem naslovu 'The Prime Pages' ('http://www.utm.edu:80/research/primes')

[2] Za funkciji $f(x)$ in $g(x)$ pomeni izraz $f(x) \sim g(x)$, da je $\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$.

DOKAZ Lastnost (i) je očitna. Edino celo število na intervalu [1, p], ki ni tuje praštevilu p, je število p. Lastnost (iii) je posledica lastnosti (ii). Slednjo lahko dokažemo z indukcijo na številu praštevil, ki delijo število n. Podroben dokaz izpustimo. Najdemo ga v [24, str. 147]. Poglejmo le primer, ki je relevanten za RSA kriptosistem. Dokažimo, da velja formula $\phi(n)$, ko je $n = pq$ produkt dveh različnih praštevil. Na intervalu [1, n] je q števil deljivih s p in p števil deljivih s q. Ker sta p in q tuji števili, je edino število, ki se ponovi, število $n = pq$. Zato je število celih števil na intervalu [1, n], ki so tuja številu n, enako

$$\phi(n) = pq - p - q + 1 = pq(1 - 1/p)(1 - 1/q) = (p - 1)(q - 1). \qquad \square$$

2.2 Števila ostankov po modulu n

Pri RSA kriptosistemu računamo s števili ostankov pri deljenju s številom n, kjer je n produkt dveh različnih praštevil. Zato bomo, podobno kot v prejšnjem razdelku, ponovili osnovne definicije in dejsteva vezana na kolobar ostankov $\mathbf{Z}_n$ po modulu n. Večina trditev nam bo prišla prav pri obravnavi RSA kriptosistema, napadov nanj in tudi implementaciji. Pri tem bomo predpostavili, da poznamo osnove algebre, ki se nanašajo na grupe, kolobarje in obsege, glej [24]. Dobra vira za števila ostankov po modulu n sta knjigi [6] in [12].

DEFINICIJA Naj bo n pozitivno celo število in a in b celi števili. Pravimo, da je *a kongruenten b po modulu n* in pišemo $a \equiv b \pmod{n}$, če $n \mid (a - b)$. Število n imenujemo *modul kongruence*. Relacijo $\equiv$ imenujemo *kongruenca*. Velja: $a \equiv b \pmod{n}$ natanko takrat, ko imata a in b enak ostanek pri deljenju s številom n.

Kongruenca je ekvivalenčna relacija pri fiksnem n. Poleg refleksivnosti, simetričnosti in tranzitivnosti za $\equiv$ veljata naslednji dve lastnosti: iz $a \equiv a_1 \pmod{n}$ in $b \equiv b_1 \pmod{n}$ sledi

$$a + b \equiv a_1 + b_1 \pmod{n} \text{ in } ab \equiv a_1 b_1 \pmod{n}.$$

Ker je za fiksen n relacija $\equiv$ ekvivalenčna, razdeli cela števila $\mathbf{Z}$ na ekvivalenčne razrede. Ekvivalenčni razred števila a je množica vseh števil kongruentih a po modulu n. Torej, če je $a = q \cdot n + r$, kjer $0 \le r \le n$, potem je $a \equiv r \pmod{n}$. Vsako število a je kongruentno nekemu številu med 0 in $n - 1$, ki ga imenujemo *najmanjši ostanek* a po modulu n. Tako sta a in r v istem ekvivalenčnem razredu, ki ga predstavlja r. Označimo $\mathbf{Z}_n = \{0, 1, \ldots, n - 1\}$.

DEFINICIJA Naj bo $a \in \mathbf{Z}_n$. *Multiplikativni inverz* števila a po modulu n je tako število $x \in \mathbf{Z}_n$, da velja $ax \equiv 1 \pmod{n}$. Če obstaja tak x, je en sam in pravimo, da je *a obrnljiv*. Pišemo a^{-1}.

DEFINICIJA Naj bosta $a, b \in \mathbf{Z}_n$. *Deljenje* a z b po modulu n je produkt števil a in b^{-1} po modulu n in je definirano samo, če je b obrnljiv po modulu n.

TRDITEV 2.9 Število $a \in \mathbf{Z}_n$ je obrnljivo natanko takrat, ko je $D(a, n) = 1$.

DOKAZ Inverz števila $a \in \mathbf{Z}_n$ obstaja, če je kongruenca $ax \equiv 1 \pmod{n}$ rešljiva v $\mathbf{Z}_n$. Slednje drži natanko takrat, ko je enačba $ax - kn = 1$ rešljiva v $\mathbf{Z}$. Ta je po trditvi 2.4 rešljiva natanko takrat, ko je $D(a, n) = 1$. $\square$

Multiplikativni inverz števila $a \in \mathbf{Z}_n$ po modulu n je rešitev enačbe $ax \equiv 1 \pmod{n}$ v $\mathbf{Z}_n$, kar ni nič drugega kot rešitev enačbe $ax + ny = 1$ v $\mathbf{Z}$. Izračunamo ga s pomočjo razširjenega Evklidovega algoritma, seveda ob pogoju, da je $D(a, n) = 1$.

TRDITEV 2.10 Naj bo $d = D(a, n)$. Kongruenčna enačba $ax \equiv b \pmod{n}$ ima rešitev $x \in \mathbf{Z}_n$ natanko tedaj, ko $d \mid b$. Rešitev je enolična po modulu n/d. Število rešitev med 0 in $n - 1$ je enako d.

DOKAZ Naj bo x_0 rešitev kongruenčne enačbe $ax \equiv b \pmod{n}$. Potem je $ax_0 - kn = b$ za nek $k \in \mathbf{Z}$. Ker $d \mid a$ in $d \mid n$, sledi $d \mid b$. Obratno, naj $d \mid b$. Z razširjenim Evklidovim algoritmom (trditev 2.3) poiščemo števili x_1 in x_2, da velja $ax_1 + nx_2 = d$. Ker je $b = dc$, za nek c, je $ax_1c + nx_2c = dc$. Izberimo $x_0 = x_1c$. Potem je $ax_0 \equiv b \pmod{n}$. Dokaz enoličnosti poteka takole. Naj bosta x_0 in x_1 dve rešitvi za enačbo $ax \equiv b \pmod{n}$. Potem je $(a/d)x_0 \equiv (a/d)x_1 \pmod{n/d}$. Ker je $D(a/d, n/d) = 1$, je po trditvi 2.9 število a/d obrnljivo. Zato lahko zadnjo kongruenco množimo z $(a/d)^{-1} \pmod{n/d}$ in dobimo $x_0 \equiv x_1 \pmod{n/d}$. Naj bo x_0 najmanjša rešitev enačbe $ax \equiv b \pmod{n}$, tista med 0 in n/d. Potem je rešitev tudi $x_k = x_0 + k(n/d)$, kjer je $1 \le k \le d - 1$. Očitno je $x_k < n$ za vsak k. □

IZREK 2.11 (Kitajski izrek o ostankih) Če so števila $n_1, n_2, \ldots, n_k$ paroma tuja, potem ima sistem kongruenčnih enačb $x \equiv a_1 \pmod{n_1}, \ldots, x \equiv a_k \pmod{n_k}$ enolično rešitev po modulu $n = n_1n_2 \cdots n_k$. Rešitev je dana s formulo

$$x = \sum_{i=1}^{k} a_i N_i M_i \bmod n,$$

kjer je $N_i = n/n_i$ in $M_i = N_i^{-1} \bmod n_i$ za $1 \le i \le k$.

DOKAZ Dokaz po Stinsonu [23, str. 119]. Za dokaz Kitajskega izreka o ostankih je dovolj pokazati, da je funkcija $\pi : \mathbf{Z}_n \to \mathbf{Z}_{n_1} \times \ldots \times \mathbf{Z}_{n_k}$, definirana s predpisom

$$\pi(x) = (x \bmod n_1, \ldots, x \bmod n_k),$$

bijektivna preslikava. Hkrati bomo na ta način dobili tudi inverzno preslikavo funkcije π. Definirajmo $N_i = n/n_i$ za vsak i, $1 \le i \le k$. Očitno je $D(n_i, N_i) = 1$. Po trditvi 2.9 obstaja multiplikativni inverz števila N_i po modulu n_i. Označimo $M_i = N_i^{-1} \bmod n_i$ za $1 \le i \le k$. Definirajmo funkcijo

$$\rho(a_1, \ldots, a_k) = \sum_{i=1}^{k} a_i N_i M_i \bmod n.$$

Potem za vsak j, $1 \le j \le k$ velja: (i) če $j = i$, potem $a_i N_i M_i \equiv a_i \pmod{n_i}$, saj je $N_i M_i \equiv 1 \pmod{n_i}$.
(ii) če $j \ne i$, potem $a_i N_i M_i \equiv 0 \pmod{n_j}$, saj $n_j \mid N_i$.

Zato je

$$X = \sum_{i=1}^{k} a_i N_i M_i \pmod{n_j} \equiv a_j \pmod{n_j}$$

za vsak j, $1 \le j \le k$, torej je X rešitev sistema kongruenčnih enačb. Ostane nam dokaz enoličnosti rešitve X po modulu n. Ker je π surjektivna funkcija na končni množici $\mathbf{Z}_n$ in velja $|\mathbf{Z}_n| = |\mathbf{Z}_{n_1} \times \ldots \times \mathbf{Z}_{n_k}|$, je tudi injektivna. Torej je π bijektivna preslikava in $\rho = \pi^{-1}$. □

POSLEDICA 2.12 Naj bo $D(n, m) = 1$. Potem imata kongruenci $x \equiv a \pmod{n}$ in $x \equiv a \pmod{m}$ enolično rešitev $x \equiv a \pmod{nm}$.

DOKAZ Obstoj in enoličnost rešitve nam da Kitajski izrek o ostankih. Ker sta n in m tuji števili, po trditvi 2.3 obstajata števili x_1 in y_1, da je $nx_1 + my_1 = 1$. Zato je $(x - a)nx_1 + (x - a)my_1 = (x - a)$. Ker $n \mid (x - a)$ in $m \mid (x - a)$, velja $(x - a) = kn = lm$ za neka $k, l \in \mathbf{Z}$. Zato $lmnx_1 + knmy_1 = (x - a)$. Očitno potem $nm \mid (x - a)$ oziroma $x \equiv a \pmod{nm}$. □

DEFINICIJA Multiplikativna grupa množice $\mathbf{Z}_n$ je $\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid D(a, n) = 1\}$. V posebnem, če je n praštevilo, potem je $\mathbf{Z}_n^* = \{a \in \mathbf{Z}_n \mid 1 \le a \le n - 1\}$. *Red* grupe $\mathbf{Z}_n^*$ je definiran kot število elementov v $\mathbf{Z}_n^*$, torej $|\mathbf{Z}_n^*|$. Iz definicije Eulerjeve funkcije sledi, da je $|\mathbf{Z}_n^*| = \phi(n)$.

IZREK 2.13 (Eulerjev izrek) Naj bo celo število $n \ge 2$. Če je $a \in \mathbf{Z}_n^*$, potem je $a^{\phi(n)} \equiv 1 \pmod{n}$.

DOKAZ Označimo $\mathbf{Z}_n^* = \{x_1, x_2, \ldots, x_{\phi(n)}\}$ in izberimo poljubno število $a \in \mathbf{Z}_n^*$. Definirajmo množico $V = \{ax_1, ax_2 \ldots, ax_{\phi(n)}\}$. Trdimo, da je $V = \mathbf{Z}_n^*$. Vzemimo poljuben $x \in \mathbf{Z}_n^*$. Potem velja $x \equiv aa^{-1}x \equiv a(a^{-1}x) \pmod{n}$, kjer $a^{-1}x \in \mathbf{Z}_n^*$, zato je $x \in V$. Torej velja $\mathbf{Z}_n^* \subseteq V$. Ker ima V največ $\phi(n)$ različnih elementov, $\mathbf{Z}_n^*$ pa natanko $\phi(n)$ različnih elementov, je $\mathbf{Z}_n^* = V$. Zato je produkt vseh elementov iz $\mathbf{Z}_n^*$ enak produktu vseh elementov iz V po modulu n:

$$x_1 \cdot x_2 \cdots x_{\phi(n)} \equiv ax_1 \cdot ax_2 \cdots ax_{\phi(n)} \pmod{n}.$$

Na desni imamo $\phi(n)$ faktorjev a. Velja $x_1x_2\cdots x_{\phi(n)} \equiv a^{\phi(n)}\, x_1x_2\cdots x_{\phi(n)} \pmod{n}$. Ker je število $x_1x_2\cdots x_{\phi(n)} \in \mathbf{Z}_n^*$, lahko z njim krajšamo levo in desno stran enačbe in dobimo $a^{\phi(n)} \equiv 1 \pmod{n}$. □

IZREK 2.14 (Fermatov izrek) Naj bo p praštevilo. Če je $D(a, p) = 1$, potem je $a^{p-1} \equiv 1 \pmod{p}$. Še več, velja $a^p \equiv a \pmod{p}$ za vsa števila $a \in \mathbf{Z}_p$.

DOKAZ Fermatov izrek je poseben primer Eulerjevega izreka, kjer $n = p$ praštevilo. Drugi del trditve velja, ker velja kongruenca $a^p \equiv a \pmod{p}$ tudi za število $0 \in \mathbf{Z}_p$. □

DEFINICIJA *Red* števila $a \in \mathbf{Z}_n^*$ je najmanjše pozitivno število t, za katerega je $a^t \equiv 1 \pmod{n}$.

TRDITEV 2.15 Naj ima $a \in \mathbf{Z}_n^*$ red t. Če za število s velja $a^s \equiv 1 \pmod{n}$, potem $t \mid s$. Posebej, red števila a deli red multiplikativne grupe $\mathbf{Z}_n^*$, torej $t \mid \phi(n)$.

DOKAZ Ker je t red $a \in \mathbf{Z}_n^*$, velja $t < s$. Recimo, da t ne deli s. Potem pri deljenje števil s in t dobimo $s = qt + r$, kjer $0 < r < t$. Velja $a^s = a^{qt+r} = (a^t)^q a^r \equiv a^r \equiv 1 \pmod{n}$. To je protislovje s predpostavko, da je t red $a \in \mathbf{Z}_n^*$. Iz Eulerjevega izreka in pravkar dokazane trditve sledi $t \mid \phi(n)$. □

TRDITEV 2.16 Naj ima $a \in \mathbf{Z}_n^*$ red t in $b \in \mathbf{Z}_n^*$ red s. Če velja $D(s, t) = 1$, potem je red ab enak st.

DOKAZ Naj ima a red t in b red s. Potem velja $(ab)^{st} \equiv 1 \pmod{n}$. Recimo, da za neko pozitivno število r velja $(ab)^r \equiv 1 \pmod{n}$. Potem je $(ab)^{rt} = a^{rt}b^{rt} \equiv b^{rt} \equiv 1 \pmod{n}$. Po trditvi 2.15 s deli rt. Ker sta s in t tuji števili, s deli r. Podobno t deli r. Potem st deli r. Torej je st red elementa ab. □

TRDITEV 2.17 Naj bo m maksimalen red števil multiplikativne grupe $\mathbf{Z}_n^*$. Potem red poljubnega števila iz $\mathbf{Z}_n^*$ deli red m.

DOKAZ Naj ima α maksimalen red m v $\mathbf{Z}_n^*$ in naj ima $a \in \mathbf{Z}_n^*$ red t. Po trditvi 2.15 števili t in m delita $\phi(n)$. Recimo, da $t \nmid m$. Potem obstaja skupni praštevilski delitelj p števil t in m, ki ima pri faktorizaciji števila t večji eksponent kot pri faktorizaciji števila m. Zato lahko zapišemo $m = p^e r$ in $t = p^f s$, kjer $f > e \ge 0$, $p \nmid r$ in $p \nmid s$. Očitno je red α^{p^e} enak r in red a^s enak p^f. Ker je $D(p^f, r) = 1$, je po trditvi 2.16 red $\alpha^{p^e} a^s$ enak $p^f r$. Toda $p^f r > m$, kar je v nasprotju z maksimalnostjo reda m. □

TRDITEV 2.18 Naj bo $D(a, n) = 1$. Potem velja $a^i \equiv a^j \pmod{n}$ natanko tedaj, ko je $i \equiv j \pmod{t}$, kjer je t red števila a.

DOKAZ Ker je $D(a, n) = 1$, je $a \in \mathbf{Z}_n^*$. Zato velja $a^{i-j} \equiv 1 \pmod{n}$. Po trditvi 2.15 velja $t \mid i - j$. Po definiciji kongruence sledi $i \equiv j \pmod{t}$. Obratno, naj velja $i \equiv j \pmod{t}$, kjer t red števila a. Tedaj je $i - j = kt$. Potem je $a^{i-j} = (a^t)^k \equiv 1 \pmod{n}$ po definciji reda. Ker je $a \in \mathbf{Z}_n^*$, velja $a^i \equiv a^j \pmod{n}$. □

TRDITEV 2.19 Če je $n \geq 2$ produkt različnih praštevil in $r \equiv s \pmod{\phi(n)}$, je $a^r \equiv a^s \pmod{n}$ za vsa števila $a \in \mathbf{Z}_n$. Posebej, če je p praštevilo in $r \equiv s \pmod{(p-1)}$, potem je $a^r \equiv a^s \pmod{p}$ za vsa števila $a \in \mathbf{Z}_n$.

DOKAZ Naj bo $a \in \mathbf{Z}_n^*$. Iz kongruence $r \equiv s \pmod{\phi(n)}$ sledi $r - s = t\phi(n)$ za nek t. Potem je

$$a^r = a^{s+t\phi(n)} = (a^{\phi(n)})^t a^s \equiv a^s \pmod{n},$$

saj je $a^{\phi(n)} \equiv 1 \pmod{n}$ po Eulerjevem izreku. Naj bo sedaj $a \in \mathbf{Z}_n \backslash \mathbf{Z}_n^*$. Ker je $n = p_1 p_2 \cdots p_k$ produkt različnih praštevil, je po trditvi 2.8, lastnosti (i) in (iii), $\phi(n) = (p_1 - 1)\cdots(p_k - 1)$. Če je $D(a, p_i) = 1$ za praštevilo p_i, $1 \leq i \leq k$, velja

$$a^r = a^{s+t\phi(n)} = (a^{p_i - 1})^{\phi_i(n)t} a^s \equiv a^s \pmod{p_i},$$

kjer $\phi_i(n) = \phi(n)/(p_i - 1)$, saj je $a^{p_i - 1} \equiv 1 \pmod{p_i}$ po Fermatovem izreku. Za $a \equiv 0 \pmod{p_i}$ očitno velja $a^r \equiv a^s \pmod{p_i}$. Po posledici 2.12 Kitajskega izreka o ostankih velja $a^r \equiv a^s \pmod{n}$ za vsa števila $a \in \mathbf{Z}_n$. □

DEFINICIJA Če ima število $\alpha \in \mathbf{Z}_n^*$ red $\phi(n)$, potem pravimo, da je α *generator* multiplikativne grupe $\mathbf{Z}_n^*$. Če ima $\mathbf{Z}_n^*$ generator, potem pravimo, da je multiplikativna grupa $\mathbf{Z}_n^*$ *ciklična.*

IZREK 2.20 (Gaussov izrek) Multiplikativna grupa $\mathbf{Z}_n^*$ ima generator natanko tedaj, ko je število n enako 2, 4, p^k ali $2p^k$, kjer je p liho praštevilo in $k \geq 1$.

DOKAZ Z direktnim poizkusom se lahko prepričamo, da ima $\mathbf{Z}_n^*$ generator, če je n enak 2 ali 4.

Naj bo $n = p$ liho praštevilo. Naj ima α maksimalen red m med vsemi števili $\mathbf{Z}_p^*$. Trdimo, da je $m = p - 1$. Po eni strani, trditev 2.15, red elementa deli red grupe, torej $m \mid (p-1)$, zato $m \leq p - 1$. Po drugi strani, gledamo rešitve enačbe $x^m - 1$ v $\mathbf{Z}_p$. Naj bo t red poljubnega elementa $a \in \mathbf{Z}_p^*$. Po trditvi 2.17 velja $t \mid m$. Zato je $m = kt$ za $k \in \mathbf{Z}$ in velja $a^m = (a^t)^k \equiv 1 \pmod{p}$ za vsa števila iz $\mathbf{Z}_p^*$. Toda enačba $x^m - 1$ ima v $\mathbf{Z}_p$ največ m različnih rešitev [24, str 176], saj je $\mathbf{Z}_p$ obseg (in zato brez deliteljev niča). Zato je $m \geq p - 1$.

Za ostale primere bomo podali le skico dokaza. Cel dokaz najdemo v [6, str. 212]. Označimo maksimalen red števil iz $\mathbf{Z}_n^*$ s $t(n)$. Najprej poiščemo vrednost $t(n)$ za števili $n = p^e$ in $n = 2^e$, kjer p liho praštevilo in $e \geq 1$. Velja

$$t(p^e) = \phi(p^e) = p^{e-1}(p-1) \text{ in } t(2^e) = 2^s,$$

kjer je $s = e - 1$, če $e = 1, 2$ in $s = e - 2$, če $e \geq 3$. Ker za število $n = ab$, kjer $D(a, b) = 1$, velja

$$t(n) = v(t(a), t(b)),$$

za splošen $n = 2^e p^{e_1} p^{e_2} \cdots p^{e_r}$, kjer so p_i liha praštevila, velja

$$t(n) = v(2^s, \phi(p^{e_1}), \phi(p^{e_2}), \cdots, \phi(p^{e_r})),$$

kjer $s = e - 1$, če $e = 1, 2$ in $s = e - 2$, če $e \geq 3$. $\mathbf{Z}_n^*$ ima generator, če je $t(n) = \phi(n)$, zato ga ima po zgornji formuli za $t(n)$ natanko tedaj, ko je $n = 2, 4, p^k$ ali $2p^k$, kjer p liho praštevilo in $k \geq 1$. □

TRDITEV 2.21 Lastnosti generatorjev multiplikativne grupe $\mathbf{Z}_n^*$:
(i) Če je α generator $\mathbf{Z}_n^*$, potem je $\mathbf{Z}_n^* = \{\alpha^i \bmod n \mid 0 \leq i \leq \phi(n) - 1\}$.
(ii) Naj bo α generator $\mathbf{Z}_n^*$. Potem je red števila $\beta = \alpha^i \bmod n$ enak $\phi(n)/D(\phi(n), i)$.
(iii) Če je $\mathbf{Z}_n^*$ ciklična, potem je število generatorjev enako $\phi(\phi(n))$.
(iv) α je generator $\mathbf{Z}_n^*$ natanko tedaj, ko je $\alpha^{\phi(n)/p} \not\equiv 1 \pmod{n}$ za vsako praštevilo p, ki deli $\phi(n)$.

DOKAZ (i) Naj bo α generator $\mathbf{Z}_n^*$. Definirajmo $a_i = \alpha^i \bmod n$, kjer $0 \le i \le \phi(n) - 1$. Potem je $a_i \in \mathbf{Z}_n^*$, saj za $x = \alpha^{\phi(n)-i} \bmod n$ očitno velja $a_i x \equiv 1 \pmod n$. Naj za indeksa i in j velja $a_i = a_j$. Potem iz $\alpha^i \equiv \alpha^j \pmod n$ po trditvi 2.18 sledi $i \equiv j \pmod{\phi(n)}$. Ker velja $0 \le i, j \le \phi(n) - 1$, je $i = j$.

(ii) Za število $\beta = \alpha^i \bmod n$, $0 \le i \le \phi(n) - 1$, je najmanjši t, za katerega je $(\alpha^i)^t \equiv 1 \pmod n$, enak $v(\phi(n), i)/i = \phi(n)/D(\phi(n), i)$.

(iii) Generatorji so vsa števila $\beta = \alpha^i \bmod n$, za katere je $D(\phi(n), i) = 1$ (glej lastnost (ii)). Torej tista števila, za katere je $i \in \mathbf{Z}_{\phi(n)}^*$. Teh je $|\mathbf{Z}_{\phi(n)}^*| = \phi(\phi(n))$.

(iv) Recimo, da obstaja praštevilo p, ki deli $\phi(n)$, za katerega velja, da je $\alpha^{\phi(n)/p} \equiv 1 \pmod n$. Tedaj je $\phi(n) = kp$ za nek $k < \phi(n)$ in $\alpha^k \equiv 1 \pmod n$. To je protislovje s predpostavko, da je število α generator $\mathbf{Z}_n^*$. Obrat je očiten, saj je potem red α enak $\phi(n)$. □

Pogosto se poleg pojma generator srečamo še s pojmom *primitivni element*. Primitivni element kolobarja $\mathbf{Z}_n$ je isto kot generator multiplikativne grupe $\mathbf{Z}_n^*$. Dovolj je vedeti, da pravzaprav govorimo o isti stvari.

TRDITEV 2.22 Število n je praštevilo natanko takrat, kadar obstaja tako število $a \in \mathbf{Z}_n$, da velja:
(i) $a^{n-1} \equiv 1 \pmod n$.
(ii) $a^{(n-1)/p} \not\equiv 1 \pmod n$ za vsako praštevilo p, ki deli $n - 1$.

DOKAZ Če je n praštevilo ima $\mathbf{Z}_n^*$ po izreku 2.20 generator. Naj bo α generator $\mathbf{Z}_n^*$. Tedaj število α zadošča (i) po definiciji generatorja in (ii) po trditvi 2.21 (iv), saj je $\phi(n) = n - 1$. Obratno, naj za $a \in \mathbf{Z}_n$ veljata predpostavki (i) in (ii). Potem ima a red $n - 1$. Po trditvi 2.15 red števila a deli red grupe $\mathbf{Z}_n^*$, ki je enak $\phi(n)$. Ker za poljubno število m velja $\phi(m) \le m - 1$, je $\phi(n) = n - 1$, kar velja le tedaj, ko je n praštevilo. □

TRDITEV 2.23 Naj bo p liho praštevilo in α generator $\mathbf{Z}_p^*$. Potem je $\alpha^{(p-1)/2} \equiv -1 \pmod p$.

DOKAZ Naj bo $u = \alpha^{(p-1)/2}$. Velja $u^2 \equiv 1 \pmod p$. Recimo, da velja $u \not\equiv -1 \pmod p$. Tedaj je $(u + 1) \in \mathbf{Z}_p^*$. Kongruenco $(u - 1)(u + 1) \equiv 0 \pmod p$ lahko množimo z $(u + 1)^{-1}$. Potem dobimo $\alpha^{(p-1)/2} = u \equiv 1 \pmod p$. Torej je red α manjši od $p - 1$, kar je protislovje, saj je α generator $\mathbf{Z}_p^*$. □

TRDITEV 2.24 Naj bodo x, y in n cela števila. Če velja $x^2 \equiv y^2 \pmod n$ in $x \not\equiv \pm y \pmod n$, potem je $D(x - y, n)$ netrivialni faktor števila n.

DOKAZ Ker število n deli $x^2 - y^2 = (x - y)(x + y)$, vendar ne deli niti $(x - y)$ niti $(x + y)$, je potem $D(x - y, n)$ netrivialni faktor za n, saj v primeru $D(x - y, n) = 1$ število n deli $(x + y)$. □

TRDITEV 2.25 Naj bo n liho praštevilo in $n - 1 = 2^s r$, kjer r liho število. Naj bo a tuje številu n. Potem velja bodisi $a^r \equiv 1 \pmod n$ bodisi $a^{2^j r} \equiv -1 \pmod n$ za nek j, $0 \le j \le s - 1$.

DOKAZ Ker je n praštevilo in $D(a, n) = 1$, velja $a^{2^s r} = a^{n-1} \equiv 1 \pmod n$ po Fermatovem izreku. Potem imamo za kongruenco $(a^{2^{s-1} r} - 1)(a^{2^{s-1} r} + 1) \equiv 0 \pmod n$ dve možnosti:

$$a^{2^{s-1} r} \equiv 1 \pmod n \text{ ali } a^{2^{s-1} r} \equiv -1 \pmod n.$$

Možnost $a^{2^{s-1} r} \not\equiv \pm 1 \pmod n$ odpade, saj bi bilo tedaj število n po trditvi 2.24 sestavljeno. Če velja prva možnost, nadaljujmo z razcepom levega faktorja. Na vsakem koraku razcepa imamo zgornji dve možnosti pri potenci $a^{2^j r}$ za nek j, $0 \le j \le s - 1$. V zadnjem koraku po trditvi 2.24 gotovo velja $a^r \equiv 1 \pmod n$. □

2.3 Teorija zahtevnosti

Glavna naloga teorije zahtevnosti je klasifikacija računskih problemov glede na vložena sredstva za dosego rešitev. Sredstva so lahko čas, prostor, število procesorjev, denar, itd. Probleme najpogosteje klasificiramo glede na vložen čas, tj. časovno zahtevnost.

DEFINICIJA *Algoritem* je računski postopek, ki nam pri danih podatkih v končnem času reši problem in vrne rezultat.

DEFINICIJA *Delovni čas* algoritma pri določenih podatkih je enak številu izvedenih *enostavnih* operacij. Največkrat enostavna operacija pomeni osnovno računsko operacijo procesorja v računalniku, npr. seštevanje, množenje, SHL (pomik v levo) ali XOR (ekskluzivni ALI).

Večkrat je težko določiti natančen delovni čas algoritma, zato si pomagamo z asimptotično oceno, kjer gledamo, kako se delovni čas algoritma povečuje s povečanjem velikosti podatkov. Za določanje časovne zahtevnosti bomo potrebovali O-notacijo, definirali pa bomo tudi o-notacijo.

DEFINICIJA *O-notacija*: $f(n) = O(g(n))$, če obstaja konstanta $c > 0$ in tako število $n_0 > 0$, da velja $0 \le f(n) \le c \cdot g(n)$ za vse $n \le n_0$. Z drugimi besedami, $f(n)$ ne raste asimptotično hitreje kot $g(n)$ do multiplikativne konstante natančno. Npr. $60x^2 + 5x + 1000 = O(x^2)$, $\sin x = O(x)$.

DEFINICIJA *o-notacija*: $f(n) = o(g(n))$, če za vsako konstanto $c > 0$, obstaja tako število $n_0 > 0$, da je $0 \le f(n) < c \cdot g(n)$ za vse $n \le n_0$. To pomeni, da je $g(n)$ zgornja meja za $f(n)$, ali drugače: $\lim_{n\to\infty} f(n)/g(n) = 0$. Npr. $10000x = o(x^2)$, $\log x = o(x)$, $x^n = o(e^x)$.

DEFINICIJA *Polinomski algoritem* je algoritem, katerega najslabši delovni čas je funkcija $O(n^k)$, kjer je n velikost podatka in k konstanta. Algoritem, katerega delovni čas ne moremo tako omejiti, imenujemo *eksponentni algoritem*.

Velikost podatkov se ponavadi meri v številu bitov, ki jih zaseda. Npr. velikost pozitivnega števila n je $\lg n = \lfloor \log_2 n \rfloor + 1$. V tem primeru je algoritem *linearen, kvadratičen* ali *polinomski* v $\lg n$, če je njegov delovni čas zaporedoma $O(\lg n)$, $O((\lg n)^2)$ ali $O(P(\lg n))$, kjer je P polinom.

Polinomskim algoritmom pravimo tudi *učinkoviti*, medtem ko eksponentnim pravimo *neučinkoviti* algoritmi. Vendar pa to v praksi ne drži vedno. Včasih se zgodi, da je eksponentni algoritem pri določenih podatkih učinkovitejši od polinomskega. Nasploh je v kriptografiji povprečni delovni čas algoritma pomembnejši od najslabšega. Npr. kriptosistem smatramo za varnega, če je ustrezen kriptoanaliticen problem v povprečju (ali še bolje, vedno) težak problem.

Teorija zahtevnosti se ukvarja predvsem z *odločitvenimi problemi*. To so problemi, za katere je rešitev problema odgovor *Da* ali *Ne*. V praksi to ne predstavlja kakšne posebne omejitve. Računski problem lahko prevedemo v primeren odločitveni problem tako, da nam učinkovite rešitve odločitvenih problemov dajo učinkovite rešitve računskih problemov in obratno.

DEFINICIJA Definirajmo naslednje razrede odločitvenih problemov:
(i) *Razred* **P** je množica odločitvenih problemov, ki so rešljivi v polinomskem času.
(ii) *Razred* **NP** je množica odločitvenih problemov, za katere lahko odgovor *Da* preverimo v polinomskem času s pomočjo neke dodatne informacije, ki pa jo je ponavadi težko pridobiti. Lahko rečemo le, če jo imamo, potem lahko učinkovito preverimo odgovor. V razredu **NP** govorimo o *nedeterministično polinomskih* odločitvenih problemih
(iii) *Razred* **co-NP** je množica odločitvenih problemov, za katere lahko odgovor *Ne* preverimo v polinomskem času z ustrezno dodatno informacijo (podobno kot pri razredu **NP**).

Za večino primerov, za katere znamo dokazati, da so v **NP** ∩ **co-NP**, se izkaže, da so v **P**. Velja **P** ⊆ **NP** in **P** ⊆ **co-NP**, še vedno pa so odprta naslednja vprašanja teorije računske zahtevnosti: ali je **P** = **NP** ∩ **co-NP,** ali je **P** = **NP**, in ali je **NP** = **co-NP**. Verjame se, da je odgovor na vsa tri vprašanja ne, čeprav tega še nikomur ni uspelo dokazati. Več o tej temi najdemo npr. v [20].

PRIMER Primer odločitvenega problema, ki je v **NP** ∩ **co-NP**, še vedno pa se ne ve ali spada v **P** ali ne, je problem SESTAVLJENOST.

Podatek: Pozitivno število n.
Vprašanje: Ali je število n sestavljeno (tj. ali obstajata števili $a, b > 1$, da je $n = ab$)?

SESTAVLJENOST spada v razred **NP**. Sestavljenost števila n lahko učinkovito preverimo, če poznamo a (dodatna informacija), delitelja n. Problem sestavljenosti števila spada tudi v **co-NP**, kajti n ni sestavljeno (je praštevilo), če poznamo število a iz trditve 2.22. Tedaj enostavno preverimo lastnosti (i) in (ii), za kar obstajajo učinkoviti algoritmi. □

DEFINICIJA Naj bosta A in B odločitvena problema. Pravimo, da se da problem A *polinomsko prevesti* na problem B (pišemo $A \leq_P B$), če obstaja algoritem, ki reši problem A, za katerega velja:
1. kot podrutino vsebuje hipotetični algoritem, ki reši problem B,
2. izvede se v polinomskem času, če se polinomsko izvede tudi algoritem za B.

Z drugimi besedami, če $A \leq_P B$, potem je problem B vsaj tako težak kot problem A, ali ekvivalentno, problem A ni nič težji od B.

DEFINICIJA Naj bosta L_1 in L_2 odločitvena problema. Če velja $L_1 \leq_P L_2$ in $L_2 \leq_P L_1$, potem pravimo, da sta problema *računsko ekvivalentna.*

TRDITEV 2.23 Za poljubne odločitvene probleme L_1, L_2 in L_3 velja naslednje:
(i) Če velja $L_1 \leq_P L_2$ in $L_2 \leq_P L_3$, potem velja $L_1 \leq_P L_3$ (tranzitivnost).
(ii) Če velja $L_1 \leq_P L_2$ in $L_2 \in$ **P**, potem velja $L_1 \in$ **P**.

DOKAZ (i) Za odločitveni problem L_1 obstaja algoritem A_1, ki reši L_1. Algoritem A_1 vsebuje hipotetični algoritem A_2, ki reši odločitveni problem L_2. Algoritem A_2 vsebuje hipotetični algoritem A_3, ki reši odločitveni problem L_3. Potem tudi algoritem A_1 vsebuje hipotetični algoritem A_3, ki reši L_3. Če se v polinomskem času izvede A_3, se v polinomskem času tudi izvede A_2 in A_1.

(ii) Za odločitveni problem L_1 obstaja algoritem A_1, ki reši L_1 in vsebuje hipotetični algoritem A_2, ki reši odločitveni problem L_2. Ker je $L_2 \in$ **P**, je po točki 2 definicije "$\leq_P$" tudi $L_1 \in$ **P**. □

Pri obravnavanju težkih problemov si večkrat pomagamo s t.i. *verjetnostnimi algoritmi,* ki uporabljajo naključna števila.

DEFINICIJA Naj bo $0 \leq \varepsilon < 1$ realno ševilo. Algoritem tipa *Las Vegas* je tak verjetnostni algoritem, da za vsak problem z verjetnostjo ε ne da odgovora (tj. vrne sporočilo "ni odgovora"), če pa ga da, je le-ta pravilen.

DEFINICIJA Naj bo $0 \leq \varepsilon < 1$ realno ševilo. Algoritem tipa *Monte Carlo* je tak verjetnostni algoritem, ki odgovori na vsak problem. Odgovor *Ne* je pravilen z verjetnostjo $1 - \varepsilon$, če pa je odgovor *Da*, je le-ta vedno pravilen.

3. O RSA

RSA kriptosistem, imenovan po avtorjih R. Rivestu, A. Shamirju in L. Adlemanu [21], je nastal leta 1977. Spada med javne kriptositeme in se v praksi največkrat uporablja za zagotavljanje zasebnosti in pristnosti podatkov. V tem poglavju bomo napisali formalno definicijo RSA kriptosistema, nato definirali problem RSA in ga povezali s problemom faktorizacije. Motivacija za to je dejstvo, da varnost RSA temelji ravno na težavnosti teh dveh računskih problemov. V preostanku poglavja bomo opisali še dve osnovni shemi, ki temeljita na RSA kriptosistemu, RSA šifrirno shemo in shemo RSA elektronskega podpisa. Ti nam bosta formalno predstavili implementacijo RSA kriptosistema v primeru šifriranja in elektronskega podpisa. Viri: [19] in [23].

3.1 RSA kriptosistem

RSA kriptositem je najbolj razširjen javni kriptosistem. Primeren je tako za uporabo v šifrirnih shemah kot tudi v shemah elektronskega podpisa. RSA kriptosistem izvaja računanje v $\mathbf{Z}_n$, kjer je modul n produkt dveh različnih praštevil p in q.

DEFINICIJA Naj bosta prostora sporočil S in šifriranih sporočil C enaka $\mathbf{Z}_n$ in prostor ključev enak

$$K = \{(n, p, q, e, d) \mid n = pq,\ p, q \text{ praštevili}, ed \equiv 1 \pmod{\phi(n)}\},$$

kjer $\phi(n) = (p - 1)(q - 1)$. Za vsak ključ $k \in K$, definiramo funkcijo šifriranja

$$E_k(x) = x^e \bmod n$$

in funkcijo dešifriranja

$$D_k(y) = y^d \bmod n,$$

kjer $x, y \in \mathbf{Z}_n$. Vrednosti n in e sta javni, vrednosti p, q, d pa zasebne. To je *RSA kriptosistem*.

Številu e pravimo *šifrirni eksponent*, številu d pa *dešifrirni eksponent*. Številu n pravimo *RSA modul*. Par (n, e) imenujemo *javni ključ*, trojico (p, q, d) pa *zasebni ključ*. Javni ključ se uporablja za šifriranje, zasebni za dešifriranje sporočil.

TRDITEV 3.1 Funkciji E_k in D_k iz definicije RSA kriptosistema sta si inverzni funkciji.

DOKAZ Ker je $ed \equiv 1 \pmod{\phi(n)}$, obstaja tako število $t \geq 1$, da je $ed = t\phi(n) + 1$. Vzemimo $x \in \mathbf{Z}_n^*$. Trdimo $D_k(E_k(x)) = x$. Velja

$$(x^e)^d = x^{t\phi(n)+1} = (x^{\phi(n)})^t\, x \equiv x \pmod{n},$$

kjer smo v zadnjem koraku uporabili Eulerjev izrek (izrek 2.13). Trditev moramo dokazati še za $x \in \mathbf{Z}_n \setminus \mathbf{Z}_n^*$. Tudi ta poteka podobno kot zgoraj, le da računamo posebej po modulu p in posebej po modulu q. Začnimo s praštevilom p. Ker je $\phi(n) = (p - 1)(q - 1)$ lahko zapišemo

$$(x^e)^d = x^{t\phi(n)+1} = (x^{p-1})^{t(q-1)}\, x \pmod{n}.$$

Če za x velja $D(x, p) = 1$, potem velja $x^{p-1} \equiv 1 \pmod{p}$ po Fermatovem izreku (izrek 2.14). Sicer $D(x, p) = p$ in velja $x \equiv 0 \pmod{p}$. V obeh primerih dobimo

$$(x^e)^d \equiv x \pmod{p}.$$

Podobno dobimo za praštevilo q enakost

$$(x^e)^d \equiv x \pmod{q}.$$

Ker sta p in q različni praštevili, je po Kitajskem izreku o ostankih (izrek 2.11) $(x^e)^d \equiv x \pmod{n}$ za vse $x \in \mathbf{Z}_n \backslash \mathbf{Z}_n^*$ in s tem za vse $x \in \mathbf{Z}_n$. □

Predpostavlja se, da je RSA kriptosistem varen, če uporabimo dovolj velik modul $n = pq$, kjer sta praštevili p in q približno enako veliki. Funkcija šifriranja $E_k(x) = x^e \bmod n$ je *enosmerna* v smislu, da je za nasprotnika dešifriranje računsko-časovno neizvedljivo brez poznavanja pravega dešifrirnega eksponenta d. Tisto, kar omogoča osebi, da dešifrira sporočilo, je poznavanje faktorjev p in q modula n. Potem lahko izračuna vrednost $\phi(n) = (p - 1)(q - 1)$ in dešifrirni eksponent d z razširjenim Evklidovim algoritmom (§5.3).

3.2 Problem RSA

Varnost mnogih kriptosistemov temelji na težavnosti različnih računskih problemov. Težavnost problema RSA je osnova za varnost RSA kriptosistema. Problem RSA je v tesni zvezi s problemom faktorizacije. Začnimo s slednjim.

DEFINICIJA Pri danem pozitivnem številu n, poišči faktorizacijo števila n na praštevila (tj. $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, kjer so p_i paroma različna praštevila in vsak $e_i \geq 1$). Ta problem imenujemo *problem faktorizacije* (FAKTORIZACIJA).

Za reševanje faktorizacijskega problema je dovolj obravnavati algoritme, ki faktorizirajo število n na dva netrivialna faktorja a in b, ki nista nujno praštevili (tj. $n = a \cdot b$, kjer $1 < a, b < n$). Ko ju najdemo, lahko testiramo praštevilskost faktorjev a in b. Isti algoritem lahko naprej uporabimo na številu, za katerega se izkaže, da je sestavljeno. Na ta način faktoriziramo število n.

Faktorizacija večjega števila ni nujno težja od faktorizacije manjšega števila. Npr. število 10^{1000} je lažje faktorizirati od števila RSA-155 (§4.1 (i)). V splošnem je težje faktorizirati število, ki ima velike faktorje, od števila, ki ima majhne faktorje.

Faktorizacija velikih števil je težak problem. Odločitveni problem, ali je število sestavljeno ali ne, se zdi v splošnem veliko bolj enostaven. Vendar je, kot smo videli v razdelku §2.3, problem SESTAVLJENOST v $\mathbf{NP} \cap \mathbf{co\text{–}NP}$. To pomeni, da trenutno ni poznan noben polinomski algoritem za rešitev tega problema.

Obstaja veliko literature za faktorizacijo in faktorizacijske algoritme. Omenimo trenutno najučinkovitejše faktorizacijske algoritme za zelo velika števila. Ti so *kvadratično rešeto* (Quadratic Sieve Algorithm) [7, str. 482], *algoritem z eliptičnimi krivuljami* (Eliptic Curve Factoring Algorithm) [7, str. 476] in *GNFS* (General Number Field Sieve) [7, str. 487]. Drugi znani algoritmi so Pollardova ρ-metoda in $p - 1$ algoritem, Williamsov $p + 1$ algoritem in seveda metoda požrešnega deljenja. Več o temi faktorizacija v [7], [19].

DEFINICIJA Pri danem pozitivnem številu n, ki je produkt dveh različnih lihih praštevil p in q, danem pozitivnem številu e, za katerega velja $D(e, (p - 1)(q - 1)) = 1$, in številu c, poišči tako število m, da je $m^e \equiv c \pmod{n}$. Ta problem imenujemo *RSA problem* (RSAP).

Z drugimi besedami, pri RSA problemu iščemo e-ti koren po modulu sestavljenega števila n. Pogoji za parametra n in e zagotavljajo, da za vsako število $c \in \{0, 1, \ldots, n-1\}$ obstaja natanko eno število $m \in \{0, 1, \ldots, n-1\}$, da je $m^e \equiv c \pmod{n}$. Z drugimi besedami, funkcija $f : \mathbf{Z}_n \rightarrow \mathbf{Z}_n$, definirana s predpisom $f(m) = m^e \bmod n$, je permutacija.

TRDITEV 3.2 RSAP $\leq_P$ FAKTORIZACIJA, ali z besedami, RSA problem lahko polinomsko prevedemo na problem faktorizacije števila.

DOKAZ Sestaviti moramo algoritem, ki reši problem RSAP in vsebuje hipotetični algoritem, ki reši problem FAKTORIZACIJA. Algoritem, ki reši problem RSAP, se mora izvesti v polinomskem času, če se v polinomskem času izvede algoritem, ki reši problem FAKTORIZACIJA.

Recimo, da imamo hipotetični algoritem A, ki reši problem faktorizacije modula n. Najprej uporabimo algoritem A, da dobimo praštevili p in q. Potem izračunamo $\phi(n) = (p-1)(q-1)$ in multiplikativni inverz števila e po modulu $\phi(n)$, $d = e^{-1} \bmod \phi(n)$, z razširjenim Evklidovim algoritmom (§5.3). Na koncu izračunamo $m = c^d \bmod n$, kar je iskana rešitev problema RSAP, saj iz $ed \equiv 1 \bmod n$ po trditvi 2.19 sledi $m^e = c^{ed} \equiv c \bmod n$. Vsi koraki se izvedejo v polinomskem času, kar bomo pokazali v poglavju §5. Zato se predstavljen algoritem izvede v polinomskem času, če se v polinomskem času izvede tudi algoritem A. □

Ali sta RSA problem in problem faktorizacije računsko ekvivalentna ni znano, znano pa je, da je problem RSAP lažji od problema FAKTORIZACIJA v primeru, ko imamo opravka z majhnim šifrirnim eksponentom e, glej [5].

3.3 RSA šifrirna shema

Najpreprostejša šifrirna shema, ki temelji na RSA kriptosistemu, je RSA šifrirna shema. Prostora sporočil M in šifriranih sporočil C sta oba enaka $\mathbf{Z}_n$, kjer je $n = pq$ produkt dveh naključno izbranih različnih praštevil. V praksi se uporablja za šifriranje krajših sporočil in prenos ključev simetričnih kriptosistemov. Varnost RSA šifrirne sheme naj bi temeljila le na težavnosti problema RSA oziroma problema faktorizacije. Algoritem 1 predstavlja način, kako priti do ključev, ki jih potrebujemo v RSA šifrirni shemi (algoritem 2).

ALGORITEM 1 - Algoritem generiranja ključev za RSA šifrirno shemo

Vsaka oseba ustvari RSA javni ključ in ustrezen zasebni ključ. Oseba A naredi naslednje:

1. Generira dve veliki praštevili
2. Izračuna $n = pq$ in $\phi(n) = (p-1)(q-1)$.
3. Izbere naključno število e, $1 < e < \phi(n)$, da velja $D(e, \phi(n)) = 1$.
4. Izračuna $d = e^{-1} \bmod \phi(n)$ z razširjenim Evklidovim algoritmom (§5.3).
5. Objavi javni ključ (n, e). Zasebni ključ osebe A je število d.

ALGORITEM 2 - RSA šifrirna shema

Oseba B zašifrira sporočilo m osebi A, ki jo le-ta dešifrira.

1. *Šifriranje*. B naj stori naslednje:
   - Dobi pristen javni ključ osebe A.
   - Reprezentira sporočilo m kot število na intervalu $[0, n-1]$.
   - Izračuna $c = m^e \bmod n$.
2. *Dešifriranje*. A pridobi prvotno sporočilo m iz šifriranega sporočila c takole:
   - Uporabi privatni ključ d in izračuna $m = c^d \bmod n$.

PRIMER Oseba A izbere praštevili $p = 101$ in $q = 113$ in izračuna števili $n = pq = 11413$ in $\phi(n) = (p-1)(q-1) = 11200$. Potem A izbere $e = 3533$, za katerega velja $D(e, \phi(n)) = 1$ (to preveri z Evklidovim algoritmom). Par (e, n) je njegov javni ključ. Nadalje izračuna $d = 6597$ z razširjenim Evklidovim algoritmom. Število d je zasebni ključ osebe A. Recimo, da želi oseba B poslati sporočilo 9726 osebi A. Pridobi javni ključ osebe A, izračuna

$$c = m^e \bmod n = 9726^{3533} \bmod 11413 = 5761$$

in ji pošlje šifrirano sporočilo 5761. Oseba A ga sprejme in z zasebnim ključem d izračuna

$$m = c^d \bmod n = 5761^{6597} \bmod 11413 = 9726. \qquad \square$$

3.4 Shema RSA elektronskega podpisa

Prostor sporočil M in prostor šifriranih sporočil C sta pri RSA kriptosistemu oba $\mathbf{Z}_n$, kjer je $n = pq$ produkt dveh naključno izbranih različnih praštevil. Ker je šifrirna transformacija bijektivna preslikava, lahko naredimo elektronski podpis z zamenjavo šifriranja in dešifriranja. Shema RSA elektronskega podpisa je shema elektronskega podpisa s sporočilom, torej pri preverjanju podpisa ne zahteva poznavanje sporočila. Prostor podpisov S je $\mathbf{Z}_n$. Funkcija $R : \mathbf{Z}_n \to M_R$, $M_R \subseteq \mathbf{Z}_n$, je javno znana injektivna funkcija, ki sporočilu ponavadi doda neko odvečno informacijo, njen inverz R^{-1} pa je lahko izračunati. Izbira primerne funkcije R je lahko ključna za varnost podpisa. Varnost predstavljene sheme temelji na težavnosti RSA problema in problema faktorizacije. V praksi se shema RSA elektronskega podpisa uporablja za majhna sporočila.

ALGORITEM 3 - Algoritem generiranja ključev za shemo RSA elektronskega podpisa

Povzetek: Oseba A ustvari javni in ustrezni zasebni ključ za RSA elektronski podpis takole:

1. Generira dve veliki praštevili, obe približno enake velikosti.
2. Izračuna $n = pq$ in $\phi(n) = (p-1)(q-1)$.
3. Izbere naključni e, $1 < e < \phi(n)$, da velja $D(e, \phi(n)) = 1$.
4. Izračuna d, $ed \equiv 1 \pmod{\phi(n)}$, $1 < d < \phi(n)$, z razširjenim Evklidovim algoritmom (§5.3).
5. Zasebni ključ osebe A je število d, javni ključ pa par (n, e).

ALGORITEM 4 - Algoritem generiranja in potrditve RSA elektronskega podpisa

Povzetek: Oseba A podpiše sporočilo $m \in M$. Oseba B lahko preveri podpis osebe A in izve sporočilo m iz podpisa.

1. *Generiranje podpisa*. Oseba A naj stori naslednje:
   - Izračuna $\tilde{m} = R(m)$, število na intervalu $[0, n-1]$.
   - Izračuna $s = \tilde{m}^d \bmod n$.
   - Podpis osebe A za sporočilo m je število s.
2. *Potrditev*. Oseba B preveri podpis s osebe A in pridobi sporočilo m takole:
   - Dobi pristen javni ključ (n, e) osebe A.
   - Izračuna $\tilde{m} = s^e \bmod n$.
   - Preveri, da je $\tilde{m} \in M_R$; če ni, podpis zavrne.
   - Pridobi sporočilo $m = R^{-1}(\tilde{m})$.

RSA kriptosistem lahko uporabimo tudi za shemo elektronskega podpisa z dodatkom. V tem primeru lahko za funkcijo R namesto injektivne funkcije uporabimo zgoščevalno funkcijo. Primer take sheme je IFSSA (§5.9), ki uporablja zgoščevalno funkcijo SHA-1 (§5.8).

4. NAPADI NA RSA

Od predstavitve RSA kriptosistema je minilo že več kot 20 let. V tem času so ga analizirali mnogi raziskovalci. Raziskave so sicer prinesle nekaj zelo zanimivih napadov, a do zdaj noben od njih ni bil usoden. Največkrat napadi prikazujejo nepravilno uporabo RSA kriptosistema. Čeprav je RSA kriptosistem relativno enostaven v primerjavi z drugimi javnimi kriptosistemi, varna implementacija le-tega nikakor ni trivialna naloga.

To poglavje obravnava različne napade na RSA kriptosistem, načine, kako se jim ubraniti, in posledično, kako povečati njegovo varnost. Opisali bomo pet razredov napadov. Pričeli bomo z napadi, ki so povezani s problemom faktorizacije. Uspešnost teh napadov bi pomenila popolni zlom RSA kriptosistema. Pred njimi se ubranimo tako, da računamo po modulu zelo velikega števila. Ker je računanje z velikimi števili časovno zelo potratno, si lahko pomagamo tako, da izberemo majhen šifrirni eksponent. Napadi na majhen šifrirni eksponent ne pomenijo popolnega zloma, kot pri faktorizaciji. Gre za dešifriranje posameznih sporočil. Tega pa ne moremo reči pri napadu na majhen dešifrirni eksponent. Majhen dešifrirni eksponent lahko povzroči popolni zlom RSA kriptosistema. Naslednji razred so napadi, ki opozarjajo na nepravilno rabo RSA kriptosistema in se v praksi ne bi smeli zgoditi. Zaključili bomo z implementacijskimi napadi. Ti ne obravnavajo RSA šifrirne funkcije kot take, pač pa skušajo zlomiti določeno vrsto implementacije RSA kriptosistema.

Najprej si na kratko poglejmo, kakšni so lahko napadi nasprotnikov in kako obravnavamo varnost. Napade nasprotnikov razdelimo v dva razreda.

(i) Pasivni napadi. V tem primeru nasprotnik spremlja le komunikacijske kanale. *Pasivni napadalec* lahko ogrozi le zaupnost podatkov.

(ii) Aktivni napadi. V tem primeru nasprotnik želi zbrisati, dodati ali kako drugače spremeniti pretok podatkov po informacijskih kanalih. *Aktivni napadalec* ogroža poleg zaupnosti še pristnost in celovitost podatkov.

Kriptosistemi z javnim ključem niso nikoli brezpogojno varni. Nasprotnik lahko ob poznavanju šifriranega sporočila c uporabi javno šifrirno funkcijo E_k za vsa možna sporočila m, dokler ne velja $c = E_k(m)$. Sporočilo m je dešifrirano sporočilo za c. Zato, ko govorimo o varnosti, ponavadi mislimo o izračunljivi varnosti določenega kriptosistema. Pri *izračunljivi varnosti* merimo količino potrebnega računskega truda za razbitje sistema po trenutno najboljših metodah. Varnost temelji na argumentih, da uspešen napad potrebuje neko stopnjo sredstev (npr. časa, prostora in denarja), ki je večja od razpoložljivih. Javni kriptosistemi navadno temeljijo na težko rešljivih računskih problemih. V primeru RSA kriptosistema je to faktorizacija. Običajno velja prepričanje, da je zahtevnost razbitja javnega kriptosistema enako težka kot razbitje ustreznega računskega problema, čeprav še ni formalnega dokaza ekvivalentnosti.

Cilj napadov na šifrirne sheme je pridobivanje sporočil iz šifriranih sporočil, ali še huje, pridobiti zasebni ključ. Večino napadov na šifrirne sheme lahko uporabimo tudi za sheme elektronskih podpisov. V tem primeru je cilj napadov najti zasebni ključ ali pa vsaj ponaredba elektronskega podpisa, tj. izdelati podpis, ki bo sprejet kot podpis neke druge osebe.

4.1 Napadi povezani s faktorizacijo

Nasprotnik želi pridobiti sporočilo m iz ustreznega šifrirnega sporočila c, pri čemer pozna javni ključ (n, e) prejemnika. To je tako imenovani RSA problem (§3.2). Trenutno ni poznan noben učinkovit algoritem za rešitev tega problema. Vemo pa, da je problem faktorizacije vsaj tako težak kot problem RSA (trditev 3.2). V tem razredu napadov na RSA kriptosistem bomo opisali napade, ki so povezani s faktorizacijo RSA modula $n = pq$, kjer sta p in q različni praštevili. Posledica faktorizacije modula je popolni zlom RSA kriptosistema, kar pomeni, da lahko za vsak šifrirni eksponent e pri danem modulu n izračunamo dešifrirni eksponent d, s tem pa lahko dešifriraramo vsa sporočila.

(i) Faktorizacija modula

Eden od očitnih napadov (vendar računsko zahtevnih) na RSA kriptosistem je, da poskusimo faktorizirati RSA modul n. Če faktorizacija modula uspe, potem mora napadalec izračunati le še $\phi(n) = (p - 1)(q - 1)$ in dešifrirni eksponent $d = e^{-1} \bmod \phi(n)$ z razširjenim Evklidovim algoritmom (glej §5.3). Ko napadalec pozna d, lahko dešifrira vsako nadaljnje sporočilo. Da bi bil RSA kriptosistem varen, je potrebno za p in q izbrati dovolj velika praštevila, da je faktorizacija $n = pq$ računsko-časovno neizvedljiva glede na trenutno tehnologijo. V praksi se dolžine modulov sučejo med 512 in 1024 biti (oz. med 155 in 310-mestno desetiško število). Trenutno so faktorizacijski algoritmi lahko faktorizirajo do 155-mestno desetiško število. Zato si je priporočljivo izbrati vsaj 100-mestna praštevila p in q, tako da je RSA modul n vsaj 200 mestno desetiško število.

Kvadratično rešeto	$O(e^{(1+o(1))\sqrt{\ln n \cdot \ln\ln n}})$
Algoritem z eliptičnimi krivuljami	$O(e^{(1+o(1))\sqrt{2\ln p \cdot \ln\ln p}})$
General Number Field Sieve	$O(e^{(1.92+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}})$

TABELA 1: Časovna zahtevnost najučinkovitejših faktorizacijskih algoritmov

Tabela 1 predstavlja računsko zahtevnost trenutno najučinkovitejših algoritmov za faktorizacijo velikih števil: kvadratično rešeto (Quadratic Sieve Algorithm), algoritem z eliptičnimi krivuljami (Eliptic Curve Factoring Algorithm) in GNFS (General Number Field Sieve) [23, str. 155]. Za faktorizacijo RSA modula ($n = pq$, kjer sta p in q približno enako veliki praštevili), je trenutno najbolj uporabna metoda GNFS. Omenimo nekatere mejnike pri faktorizaciji velikih števil. Leta 1983 so s pomočjo kvadratičnega rešeta Davis, Holdredge in Simmons uspešno faktorizirali število $2^{251} - 1$, katerega faktor je bilo (sestavljeno) 69-mestno število. Do leta 1989 sta Lanstra in Manasse uspela faktorizirati že 106-mestni faktor. Leta 1994 so Atkins, Graff, Lenstra in Leyland uspeli faktorizirati 129-mestni faktor, znan kot RSA-129[1]. Pri faktorizaciji RSA-129 s kvadratičnim rešetom je sodelovalo preko 600 udeležencev po vsem svetu. Števila RSA-100, RSA-110, …, RSA-500 so skupina RSA modulov, ki so podani na internetu[2] kot izziv za faktorizacijske algoritme. GNFS je najnovejša metoda za faktorizacijo. Izgleda, da ima velik potencial, saj je hitrejša od drugih dveh. Za ta algoritem se predvideva, da je hitrejši za števila večja od 130-mest. Leta 1990 je ta algoritem faktoriziral $2^{2^9} - 1$ v tri števila, dolga 7, 49 in 99 mest. Aprila 1996 je sledila faktorizacija števila RSA-130, februarja 1999 pa se je po devetih tednih uspešno končala faktorizacija števila RSA-140 [8].

[1] RSA-d je d-mestno desetiško število, ki je produkt dveh praštevil, približno enake velikosti.
[2] Več o faktorizacijskih izzivih na spletnem naslovu http://www.rsasecurity.com/rsalabs/challenges/

Najpomembnejši rezultat do zdaj je faktorizacija števila RSA-155[3], ki je bila končana avgusta 1999 po sedmih mesecih. Skupina ljudi na čelu z Lenstra in Riele so opravili potrebene izračune na 300 računalnikih. Rezultat nam pove, da lahko dobro organizirana skupina nasprotnikov zlomi 512 bitni ključ le v nekaj tednih. Vendar pa je cena za zlom 512-bitnega ključa še vedno dovolj velika, da prepreči potencialnim napadalcem implementacijo napada v širšem smislu.

(ii) Izračun dešifrirnega eksponenta

Faktorizacija RSA modula n je eden od možnih pristopov, ki ga lahko nasprotnik uporabi za rešitev problema RSA. Potem izračuna $\phi(n)$ in dešifrirni eksponent d tako, kot bi to naredil prejemnik sporočila. Po drugi strani, če je nasprotniku nekako znan d, potem lahko, kot bomo videli, faktorizira n.

Napisali bomo verjetnostni algoritem, ki faktorizira RSA modul n tako, da uporabi za podrutino algoritem, ki izračuna dešifrirni eksponent d. Recimo, da je A hipotetični algoritem, ki izračuna d iz šifrirnega eksponenta e. Opisali bomo Las Vegas algoritem (glej §2.3), ki uporabi A kot podrutino. Ta algoritem bo faktoriziral n z verjetnostjo vsaj $\varepsilon = 1/2$. Če ta algoritem poženemo m-krat, bomo n faktorizirali z verjetnostjo vsaj $1 - (1/2)^m$.

Algoritem temlji na dejstvu, ki se tičejo kvadratnih korenov za 1 po modulu n, kjer je $n = pq$ produkt dveh lihih praštevil. Ker velja $x^2 \equiv 1 \pmod{n}$ natanko takrat, ko je $x^2 \equiv 1 \pmod{p}$ in $x^2 \equiv 1 \pmod{q}$, sledi, da je $x^2 \equiv 1 \pmod{n}$ natanko takrat, ko $x = \pm 1 \bmod p$ in $x = \pm 1 \bmod q$. Zato imamo štiri kvadratne korene za 1 po modulu n, ki jih poiščemo s Kitajskim izrekom o ostankih (izrek 2.11). Dve od teh rešitev sta trivialni, $x = \pm 1 \bmod n$. Drugi dve rešitvi sta $\pm x$, kjer x zadošča $x = 1 \bmod p$ in $x = -1 \bmod q$ (*netrivialna kvadratna korena* za 1 po modulu n).

Naj bo x netrivialni kvadratni koren za 1 po modulu n. Potem $n \mid (x - 1)(x + 1)$, toda n ne deli noben faktor na desni strani. Sledi, da je $D(x + 1, n) = p$ (ali q). Podobno je $D(x - 1, n) = q$ (ali p). Seveda izračunamo največji skupni delitelj s pomočjo Evklidovega algoritma. Torej, poznavanje netrivialnega kvadratnega korena za 1 po modulu n pripelje do faktorizacije n v polinomskem času.

ALGORITEM 5 - Faktorizacijski algoritem pri danem dešifrirnem eksponentu d

Podatki: javni ključ (n, e) in dešifrirni eksponent d.
Rezultat: praštevili p in q, v primeru uspeha.

1. izberi naključno število w, kjer $1 \le w \le n - 1$.
2. $x := D(w, n)$;
3. if $(1 < x < n)$ then return (x); (uspeh: $x = p$ ali $x = q$)
4. $d := A(e)$;
5. zapiši $ed - 1 = 2^s \cdot r$, kjer je r liho število
6. $v := w^r \bmod n$;
7. if $(v \equiv 1 \pmod{n})$ then quit; (neuspeh)
8. while $(v \not\equiv 1 \pmod{n})$ do
   8.1 $v_0 := v$;
   8.2 $v := v^2 \bmod n$;
9. if $(v_0 \equiv -1 \pmod{n})$ then quit; (neuspeh)
   else $x := D(v_0 + 1, n)$, return (x); (uspeh: $x = p$ ali $x = q$)

[3] Več o faktorizaciji RSA-155 na spletnem naslovu 'http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html'

Analizirajmo algoritem 5. Sledimo dokazu Stinsona [23, str. 139]. Najprej opazimo, da če izberemo w, ki je večkratnik p ali q, potem takoj faktoriziramo n (koraku 2). Če sta w in n tuja, potem izračunamo w^r, w^{2r}, w^{4r}, … z zaporednim kvadriranjem, dokler ne velja

$$w^{2^t r} \equiv 1 \pmod{n}$$

za neko število t. Ker je $ed - 1 = 2^s r \equiv 0 \pmod{\phi(n)}$, je $w^{2^s r} \equiv 1 \pmod{n}$. Zato bo imela *while* zanka največ s iteracij (korak 8). Na koncu *while* zanke smo našli tako število v_0, da velja $v_0^2 \equiv 1 \pmod{n}$ in $v_0 \not\equiv 1 \pmod{n}$. Če je $v_0 \equiv -1 \pmod{n}$, potem algoritem ni uspel, sicer je v_0 netrivialni kvadratni koren za 1 po modulu n in lahko faktoriziramo n (korak 9).

Naš cilj je pokazati, da algoritem uspe z verjetnostjo najmanj 1/2. Imamo dva primera, v katerih algoritem ne uspe faktorizirati n:

1. $w^r \equiv 1 \pmod{n}$
2. $w^{2^t r} \equiv -1 \pmod{n}$ za nek t, $0 \le t \le s - 1$

Če je w rešitev najmanj ene od teh $s + 1$ kongruenc, potem je w slaba izbira. Preštejmo število rešitev za vsako od teh kongruenc posebej. Začnimo z $w^r \equiv 1 \pmod{n}$. Gledamo rešitve po modulu p in q, potem pa jih združimo s Kitajskim izrekom o ostankih (izrek 2.11). Velja:

$$w^r \equiv 1 \pmod{n} \iff w^r \equiv 1 \pmod{p} \text{ in } w^r \equiv 1 \pmod{q}.$$

Najprej imejmo kongruenco $w^r \equiv 1 \pmod{p}$. Ker je p praštevilo, je po izreku 2.20 $\mathbf{Z}_p^*$ ciklična grupa. Naj bo α generator $\mathbf{Z}_p^*$. Zapišemo lahko $w = \alpha^\mu$ za nek u, $0 \le u \le p - 2$. Iz $w^r \equiv 1 \pmod{p}$ sledi $\alpha^{\mu r} \equiv 1 \pmod{p}$. Po trditvi 2.18 sledi $ur \equiv 0 \pmod{(p-1)}$ oziroma $(p-1) \mid u \cdot r$.

Zapišimo $p - 1 = 2^i p_1$ in $q - 1 = 2^j q_1$, kjer sta p_1 in q_1 lihi števili. Ker $\phi(n) = (p-1)(q-1)$ deli $(e \cdot d - 1) = 2^s r$, velja $2^{i+j} p_1 q_1 \mid 2^s r$. Od tod sledi $(i + j) \le s$ in $p_1 q_1 \mid r$. Pogoj $(p-1) \mid u \cdot r$ postane pogoj $2^i p_1 \mid u \cdot r$. Ker $p_1 \mid r$, kjer r lih, je potrebno in zadostno, da $2^i \mid u$. Od tod sledi, da je $u = k \cdot 2^i$, $0 \le k \le p_1 - 1$, število rešitev kongruence $w^r \equiv 1 \pmod{p}$ pa je p_1.

Podobno je število rešitev kongruence $w^r \equiv 1 \pmod{q}$ enako q_1. S Kitajskim izrekom o ostankih lahko združimo katerikoli rešitvi po modulu p in q v enolično rešitev po modulu n. Torej je število rešitev kongruence $w^r \equiv 1 \pmod{n}$ enako $p_1 \cdot q_1$.

Naslednji korak je kongruenca $w^{2^t r} \equiv -1 \pmod{n}$ za fiksno vrednost t, kjer $0 \le t \le s - 1$. Kot prej rešujemo kongruenco po modulu p in q. Spet velja:

$$w^{2^t r} \equiv -1 \pmod{n} \iff w^{2^t r} \equiv -1 \pmod{p} \text{ in } w^{2^t r} \equiv -1 \pmod{q}.$$

Za primer $w^{2^t r} \equiv -1 \pmod{p}$ zapišemo $w = \alpha^\mu$ kot prej in dobimo $\alpha^{u 2^t r} \equiv -1 \pmod{p}$. Ker je po trditvi 2.23 $\alpha^{(p-1)/2} \equiv -1 \pmod{p}$, je $u 2^t r \equiv (p-1)/2 \pmod{(p-1)}$ po trditvi 2.18. Torej $(p-1)$ deli $(u 2^t r - (p-1)/2)$. Spet pišemo $p - 1 = 2^i p_1$ in (po množenju z 2) dobimo $2^{i+1} p_1 \mid (u 2^{t+1} r - 2^i p_1)$. Delimo to še s p_1 in dobimo $2^{i+1} \mid ((u 2^{t+1} r / p_1) - 2^i)$. Za primer $t \ge i$ ni rešitev, saj $2^{i+1} \mid 2^{t+1}$, toda $2^{i+1} \nmid 2^i$. Po drugi strani za primer $t \le i - 1$ velja, da je $u \in \{0, 1, \ldots, p-1\}$ rešitev natanko takrat, ko je u lih večkratnik 2^{i-t-1} (saj je r/p_1 je liho število). Torej je število rešitev v tem primeru enako $(p-1)/2^{i-t-1} \cdot 1/2 = 2^t \cdot p_1$.

Podobno velja za kongruenco $w^{2^t r} \equiv -1 \pmod{q}$. Za $t \ge j$ nima rešitev, za $t \le j - 1$ pa ima $2^t \cdot q_1$ rešitev. Po Kitajskem izreku o ostankih je število rešitev za kongruenco $w^{2^t r} \equiv -1 \pmod{n}$ enako 0, če $t \ge \min\{i, j\}$, in $2^{2t} p_1 \cdot q_1$, če $t \le \min\{i, j\} - 1$.

Sedaj pa še preštejmo vse rešitve obeh primerov. Upoštevati moramo, da velja $0 \le t \le s - 1$. Brez škode za splošnost, predpostavimo, da je $i \le j$. Potem je število rešitev 0 za $t \ge i$. Število vseh "slabih" izbir za w je največ

$$p_1 \cdot q_1 + p_1 \cdot q_1(1 + 2^2 + 2^4 + \ldots + 2^{2i-2}) = p_1 \cdot q_1(1 + (2^{2i} - 1)/3) = p_1 \cdot q_1(2/3 + 2^{2i}/3).$$

Spomnimo se, da je $p - 1 = 2^i p_1$ in $q - 1 = 2^j q_1$. Ker je $j \ge i \ge 1$, velja $p_1 \cdot q_1 < n / 4$. Prav tako velja $2^{2i} p_1 \cdot q_1 \le 2^{i+j} \cdot p_1 \cdot q_1 = (p - 1)(q - 1) < n$. Torej je število rešitev

$$p_1 \cdot q_1(2/3 + 2^{2i}/3) < n/6 + n/3 = n/2.$$

Ker je kvečjemu $(n - 1)/2$ slabih izbir za w, sledi, da je vsaj $(n - 1)/2$ dobrih izbir za w in zato je verjetnost za uspeh algoritma vsaj 1/2.

TRDITEV 4.1 Izračun dešifrirnega eksponenta d iz javnega kjuča (n, e) in problem faktorizacije n sta računsko enako zahtevna.

DOKAZ Ker je $ed \equiv 1 \pmod{\phi(n)}$, velja $ed - 1 = k \cdot \phi(n)$ za nek k. Po Eulerjevem izreku je $a^{ed-1} = 1 \pmod n$ za vsak $a \in \mathbf{Z}_n^*$. Naj bo $ed - 1 = 2^s r$, kjer je r liho število. Potem obstaja tako število i, $1 \le i \le s$, da velja

$$a^{2^{i-1} r} \not\equiv \pm 1 \pmod n \quad \text{in} \quad a^{2^i r} \equiv 1 \pmod n$$

za vsaj polovico vseh $a \in \mathbf{Z}_n^*$ (glej analizo algoritma 5). Če sta a in i taki števili, je $D(a^{2^{i-1} r} - 1, n)$ netrivialni faktor za n. Tako lahko nasprotnik enostavno ponavlja izbor $a \in \mathbf{Z}_n^*$ in preverja, ali za katerega od i, $1 \le i \le s$, velja zgornja lastnost. Pričakovano število poskusov za izračun netrivialnega faktorja modula n je 2. □

Ta rezultat je pomemben. Pove nam, če je d odkrit, potem je tudi n ogrožen. Torej, če se to zgodi, ni zadosti zamenjati dešifrirni eksponent d, ampak tudi modul n.

(iii) Izračun $\phi(n)$

Poglejmo si, da je za uspešen napad dovolj izračunti $\phi(n)$. Če sta števili n in $\phi(n)$ znani, potem lahko n faktoriziramo, če rešimo ta sistem enačb:

$$n = p \cdot q$$
$$\phi(n) = (p - 1)(q - 1)$$

za neznana p in q. V drugo enačbo vstavimo $q = n / p$ in dobimo kvadratno enčbo za p:

$$p^2 - (n - \phi(n) + 1) \cdot p + n = 0.$$

Rešitvi te kvadratne enačbe bosta p in q, faktorja pri n. Tako lahko nasprotnik, če izve $\phi(n)$, faktorizira n in razbije sistem. Z drugimi besedami:

TRDITEV 4.2 Izračun $\phi(n)$ in problem faktorizacije n sta računsko enako zahtevna.

PRIMER Recimo, da nasprotnik pozna števili $n = 84773093$ in $\phi(n) = 84754668$. Potem mora rešiti naslednjo kvadratno enačbo

$$p^2 - 18426p + 84773093 = 0.$$

Rešitvi kvadratne enačbe 9539 in 8887 sta natanko faktorja modula n. □

4.2 Majhen šifrirni eksponent

Za učinkovito šifriranje je dobro izbrati majhen šifrirni eksponent ali pa šifrirni eksponent, ki ima v binarni reprezentaciji malo enic. Najmanjša možna vrednost za šifrirni eksponent je 3, vendar je za uspešno obrambo pred določenimi napadi priporočena vrednost $e = 2^{16} + 1 = 65537$. Napadi na majhen šifrirni eksponent ne pomenijo popolnega zloma kot pri faktorizaciji.

Najnevarnejši napadi pri majhnem šifrirnem eksponentu temeljijo na izreku Coppersmitha [9], ki ga bomo predstavili brez dokaza, saj zaradi obsežnosti in zapletenosti presega obseg diplomske naloge. Pojasnimo le nekatere pojme, s katerimi se bomo srečali v tem razdelku. Množico polinomov s koeficienti v $\mathbf{Z}_n$ označimo z $\mathbf{Z}_n[x]$. Stopnjo polinoma $f \in \mathbf{Z}_n[x]$ označimo z $\deg(f)$. Če ima polinom vodilni koeficient enak 1, ga imenujemo *moničen polinom*. V kolobarju polinomov $\mathbf{Z}_n[x]$ imamo analogne definicije in izreke kot pri teoriji števil (glej §2.1 in §2.2), npr. izrek o deljenju, izrek o enolični faktorizaciji, običajen in razširjen Evklidov algoritem, Kitajski izrek o ostankih, itd (glej [6], [19], [24]). Nekaj teh bomo v nadaljevanju tudi uporabili.

IZREK 4.3 (Coppersmith) Naj bo n celo število in $f \in \mathbf{Z}_n[x]$ moničen polinom stopnje δ. Potem lahko pri danih n in f učinkovito poiščemo vsa števila $|x_0| < n^{1/\delta}$, ki zadoščajo enačbi

$$f(x_0) \equiv 0 \pmod{n},$$

v času, ki je polinomski v 2^δ in $\lg n$.

Coppersmith [9] opiše učinkovit algoritem za iskanje vseh ničel polinoma $f \in \mathbf{Z}_n[x]$ po modulu n, ki so manjša od $n^{1/\delta}$. Moč izreka je v tem, da lahko išče majhne ničle polinomov po modulu sestavljenega števila n. Ko delamo s praštevilskim modulom, ni potrebe, da bi za iskanje ničel uporabili Coppersmithov izrek, saj obstajajo veliko boljši algoritmi [2].

(i) Širokopasovni napad

Hastad [13] je našel naslednji napad na RSA kriptosistem pri majhnem šifrirnem eksponentu. Recimo, da želimo poslati isto sporočilo m različnim osebam $A_1, A_2, \ldots, A_k$. Vsaka oseba ima svoj javni ključ (n_i, e_i). Osebe A_i imajo lahko enak šifrirni eksponent, toda vsaka oseba ima drug modul. Predpostavimo lahko, da so moduli paroma tuji, sicer lahko nasprotnik z Evklidovim algoritmom poišče $D(n_i, n_j)$ tistih modulov n_i in n_j, ki niso tuji, in jih tako faktorizira. Predpostavimo še, da velja $m < n_i$ za vse i.

Poglejmo si enostaven primer takega napada. Recimo, da imajo vse osebe enak šifrirni eksponent $e = 3$. Videli bomo, da lahko nasprotnik izračuna sporočilo m, če je $k \geq 3$. Če želimo poslati isto sporočilo m trem različnih osebam, ki imajo javne module n_1, n_2, n_3 in šifrirni eksponent $e = 3$, potem bomo poslali šifrirana sporočila

$$c_i = m^3 \bmod n_i, \quad i = 1, 2, 3.$$

Ker so moduli n_i paroma tuji, lahko nasprotnik s Kitajskim izrekom o ostankih (izrek 2.11) poišče rešitev x, $0 \leq x < n_1 \cdot n_2 \cdot n_3$, za katero velja

$$x = m^3 \bmod n_1 \cdot n_2 \cdot n_3.$$

Ker je $m < n_i$ za vse i, je $m^3 < n_1 \cdot n_2 \cdot n_3$. Zato velja enakost $x = m^3$ tudi v $\mathbf{Z}$. Nasprotnik lahko izračuna m tako, da izračuna kubični koren števila x. Enak razmislek velja za poljuben šifrirni eksponent e. V primeru, da ima k oseb enak šifrirni eksponent in velja $m < n_i$ za vse i, $1 \leq i \leq k$, lahko nasprotnik izračuna sporočilo m, če je $k \geq e$. Ker v praksi ponavadi ne pošiljamo isto šifrirano sporočilo večji skupini ljudi, je napad uporaben le za majhen šifrirni eksponent.

Hastad je opisal še močnejši napad [13]. Naivna obramba pred zgornjim napadom bi bila vsakemu sporočilu dodati neko informacijo. Npr. če je dolžina sporočila m enaka l bitov, potem lahko vsaki osebi A_i pošljemo sporočilo $m_i = i \cdot 2^l + m$ (z drugimi besedami pred m dodamo še i). Tako bi vsaki osebi poslali drugačno sporočilo. Toda Hastad je pokazal, da tako dodajanje informacij tudi ni povsem varno.

Recimo, da za vsako izmed oseb $A_1, \ldots, A_k$ določimo in objavimo polinom $f_i \in \mathbf{Z}_{n_i}[x]$. Vsaki osebi A_i pošljemo sporočilo $f_i(m)$. Nasprotnik prestreže šifrirana sporočila

$$c_i = f_i(m)^e \bmod n_i, \quad i = 1, \ldots, k.$$

Hastad je pokazal, da lahko nasprotnik pridobi sporočilo m iz vseh šifriranih sporočil c_i, če je pri tem udeleženo dovolj oseb. Naslednji izrek [2] je močnejša različica Hastadovega originalnega izreka.

IZREK 4.4 (Hastad) Naj bodo $n_1, \ldots, n_k$ paroma tuja števila in n najmanjši med njimi. Naj bodo $g_i \in \mathbf{Z}_{n_i}[x]$, $i = 1, \ldots, k$, polinomi stopnje največ δ. Naj obstaja število $m < n$, za katero velja

$$g_i(m) \equiv 0 \pmod{n_i}, \quad i = 1, \ldots, k.$$

Če pri danih n_i in g_i velja $k \geq \delta$, potem lahko učinkovito poiščemo število m.

DOKAZ Naj bo $N = n_1 \cdot n_2 \cdots n_k$. Če vodilni koeficient polinom g_i ni obrnljiv element v $\mathbf{Z}_{n_i}^*$, potem lahko faktoriziramo n_i. Če polinom g_i ni stopnje δ, ga pomnožimo s primerno potenco x, da postane njegova stopnja enaka δ. Torej lahko predpostavimo, da so g_i monični polinomi stopnje δ. Konstruirajmo polinom

$$g(x) = \sum_{i=1}^{k} T_i g_i(x),$$

kjer so števila T_i koeficienti rešitve iz Kitajskega izreka o ostankih. Potem ima polinom $g(x)$ vodilni koeficient enak $\sum_{i=1}^{k} T_i$. Ker velja $\sum_{i=1}^{k} T_i \equiv 1 \pmod{n_j}$ za vsak j, je $\sum_{i=1}^{k} T_i \equiv 1 \pmod{N}$ po posledici 2.12. Zato je $g(x)$ moničen polinom stopnje δ in zadošča enačbi $g(m) \equiv 0 \pmod{N}$. Ker je $m < n < N^{1/k} < N^{1/\delta}$ lahko po izreku 4.3 učinkovito poiščemo število m. □

Izrek nam pove, da lahko učinkovito rešimo sistem enačb ene spremenljivke po modulu tujih si števil, če le imamo dovolj enačb. Če določimo $g_i(x) = (f_i^{e_i}(x) - c_i) \bmod n_i$, potem lahko nasprotnik pridobi sporočilo m iz danih šifriranih sporočil, če je število oseb vsaj δ, kjer je

$$\delta = \max\{e_i \deg(f_i) \mid i = 1, \ldots, k\}.$$

V posebnem primeru, ko so vsi e_i enaki e in pošiljamo linearno sorodna sporočila, potem lahko nasprotnik pridobi sporočilo m za $k \geq e$.

Majhen šifrirni element ne smemo uporabiti v primeru, ko pošiljamo isto sporočilo več uporabnikom hkrati. Za obrambo pred tem napadom moramo sporočilu pred šifriranjem na koncu dodati naključen niz bitov primerne dolžine, npr. najmanj 64 bitov, ali pa kar tako velik niz, da je dolžina niza skupaj s sporočila reda velikosti dolžine modula. Ta niz mora biti neodvisno generiran za vsako sporočilo posebej. Tej metodi pravimo *soljenje sporočil* (angl. salting messages).

(ii) Franklin-Reiterjev napad

Franklin in Reiter [10] sta našla napad pri majhnem šifrirnem eksponentu pri pošiljanju *sorodnih sporočil* z istim modulom. Recimo, da imamo dve različni sporočili m_1 in m_2, ki zadoščata relaciji

$$m_2 = \alpha m_1 + \beta .$$

Naj bo šifrirni eksponent e enak 3 in naj bosta $c_i = m_i^3 \bmod n$, $i = 1, 2$, ustrezni šifrirani sporočili. Potem lahko iz $(c_1, c_2, \alpha, \beta, n)$ eksplicitno izračunamo sporočili m_1 in m_2 na sledeč način:

$$\frac{\beta(c_2 + 2\alpha^3 c_1 - \beta^3)}{\alpha(c_2 - \alpha^3 c_1 + 2\beta^3)} = \frac{3\alpha^3\beta\, m_1^3 + 3\alpha^2\beta^2\, m_1^2 + 3\alpha\beta^3\, m_1}{3\alpha^3\beta\, m_1^2 + 3\alpha^2\beta^2\, m_1 + 3\alpha\beta^3} = m_1 \bmod n,$$

sporočilo m_2 pa izračunamo iz linearne zveze med m_1 in m_2. Ideja je ta, da poiščemo polinoma $P(m)$ in $Q(m)$, ki zadoščata enačbi $Q(m) = mP(m)$. Podobno lahko naredimo za poljuben šifrirni eksponent e in dve linearno povezani sporočili. Vendar je ta naloga za večje e precej bolj zahtevna. Na srečo imamo enostavnejšo metodo.

Naj bo (n, e) javni ključ. Recimo, da sta $m_1, m_2 \in \mathbf{Z}_n^*$ dve različni sporočili, ki zadoščata relaciji

$$m_2 = f(m_1) \bmod n,$$

kjer je $f \in \mathbf{Z}_n[x]$ javno znan linearen polinom. Sporočili šifriramo v $c_i = m_i^e \bmod n$, $i = 1, 2$, in ju pošljemo. Pri danih šifriranih sporočilih c_1 in c_2 ter šifrirnem eksponentu e sta enačbi

$$x^e - c_1 \equiv 0 \pmod n$$
$$f(x)^e - c_2 \equiv 0 \pmod n$$

rešljivi za $x = m_1$ in zato je $D(x^e - c_1, f(x)^e - c_2) \in \mathbf{Z}_n[x]$. Razen v izjemnih primerih [10] bo veljalo

$$D(x^e - c_1, f(x)^e - c_2) = x - m_1.$$

Napad lahko izvedemo pri poljubnem šifrirnem eksponentu e, vendar je ta omejen z zahtevnostjo izračuna največjega skupnega delitelja dveh polinomov stopnje e. Izračun lahko opravimo z Evklidovim algoritmom za polinome [19, str. 85] v času $O(e^2(\lg n)^2)$. Napad je uporaben za majhne šifrirne eksponente z dolžino do 32 bitov. Npr. napad je učinkovit za $e = 2^{16} + 1 = 65537$, ki je pogosta izbira za šifrirni eksponent. Napad lahko posplošimo na polinome poljubne stopnje in na poljubno število sorodnih sporočil. [10].

Očitna obramba pred zgornjim napadom je povečanje šifrirnega eksponenta e, vendar s tem izgubimo na hitrosti šifriranja oziroma preverjanja podpisa. Druga možnost obrambe je soljenje sporočil, ki smo ga opisali v 4.2 (i). Vendar, kot bomo videli pri naslednjem napadu, ta metoda ni vedno zadovoljiva.

(iii) Napad na soljena sporočila

Franklin-Reiterjev napad (§4.2 (ii)) se morda ne zdi naraven. Le zakaj bi hoteli pošiljati sorodna sporočila. Coppersmith [9] je izboljšal napad in s tem podal pomemben rezultat pri metodi soljenja sporočil (glej §4.2 (i)).

Spomnimo se napada Franklina in Reiterja. Recimo, da imamo dve sorodni sporočili m in m', tj. taki sporočili, ki zadoščata znani relaciji, npr. $m' = m + r$, kjer je r znan. Naj bo šifrirni eksponent e enak 3 in recimo, da poznamo šifrirani sporočili

$$c = m^3 \bmod n \text{ in } c' = (m')^3 \bmod n.$$

Potem lahko z Franklin-Reiterjevim napadom iz števil c, c', r in n izračunamo sporočilo m. Kaj pa, če ne poznamo natančno relacije med m in m', toda vemo, da je r majhen, npr. $|r| < n^{1/9}$. Ali lahko poiščemo sporočilo m?

Naiven algoritem za soljenje sporočil doda na konec sporočila nekaj naključnih bitov. Coppersmith nas opozori na nevarnost takega pristopa. Poglejmo za kaj gre. Kot prej, naj bo $e = 3$. Naj bo soljeno sporočilo m enako

$$m = 2^k M + R,$$

kjer je R enak k naključno dodanim bitom. Sporočilo m šifriramo in ga pošljemo. Nasprotnik prestreže to sporočilo. Ker nismo dobili odgovora, spet dodamo k naključnih bitov R' k sporočilu M. Soljeno sporočilo

$$m' = 2^k M + R'$$

šifriramo in ga spet pošlejemo. Nasprotnik ima sedaj dve šifrirani sporočili istega sporočila M z dodanimi naključnimi biti. Recimo, da je $R' = R + r$ in zato $m' = m + r$. Potem imamo

$$c = m^3 = (2^k M + R)^3 \bmod n$$
$$c' = (m')^3 = (2^k M + R')^3 = (m + r)^3 \bmod n.$$

Ali lahko dobimo r in m, če poznamo c, c' in n? Odgovor nam da naslednja trditev.

TRDITEV 4.5 (Coppersmith) Naj bo (n, e) javni ključ. Naj bo $M \in \mathbf{Z}_n^*$. Definirajmo

$$m_1 = 2^k M + r_1 \text{ in } m_2 = 2^k M + r_2,$$

kjer sta r_1 in r_2 različni števili, katerih dolžina k je manjša od $(\lg n)/e^2$. Naj bosta c_1, c_2 ustrezni šifrirani sporočili za sporočili m_1, m_2. Tedaj lahko pri danih (n, e, c_1, c_2) učinkovito poiščemo m.

DOKAZ Recimo, da je $r_2 = r_1 + r$. Potem je $m_2 = m_1 + r$. Definirajmo

$$g_1(x, y) = x^e - c_1 \text{ in } g_2(x, y) = (x + y)^e - c_2.$$

Polinoma g_1 in g_2 imata pri $y = r_2 - r_1$ skupno rešitev m_1. Ali drugače, $r = r_2 - r_1$ je rešitev polinoma $h(y) = R(g_1, g_2) \in \mathbf{Z}_n[y]$ stopnje največ e^2, kjer je $R(g_1, g_2)$ rezultanta polinomov g_1 in g_2, glej [24, str. 194]. Poleg tega velja $|r| < 2^k < n^{1/e^2}$. Zato lahko na polinomu h uporabimo izrek 4.3 in poiščemo vrednost r. Ko poznamo r, uporabimo Franklin-Reiterjev napad (§4.2 (ii)) in poiščemo rešitev m_1. Zatem odstranimo naključne bite in dobimo sporočilo M. □

Nadaljujmo prejšnji primer. Po dokazu trditve 4.5 moramo izračunati rezultanto polinomov $m^3 - c$ in $(m + r)^3 - c'$. Z njo se znebimo spremenljivke m.

$$R(m^3 - c, (m + r)^3 - c') = r^9 + (3c - 3c')r^6 + (3c^2 + 21cc' + 3(c')^2)r^3 + (c - c')^3 \equiv 0 \pmod n.$$

To je polinom ene spremenljivke r stopnje 9 po modulu n. Če velja $|r| < n^{1/9}$, lahko uporabimo izrek 4.3 in dobimo vrednost r. Potem uporabimo Franklin-Reiterjev napad, da dobimo m. Na koncu odstranimo naključne bite, da dobimo M. Napad lahko izvedemo tudi, če dodamo naključne bite na začetek ali sredino sporočila, glej [9]. Skratka, napad na soljena sporočila pri šifrirnem eksponentu $e = 3$ lahko izvedemo, če je dolžina naključno dodanih bitov manjša od 1/9 dolžine modula n. Napad ni uporaben za večje e, npr. $e = 2^{16} + 1 = 65537$.

(iv) Napad na stereotipna sporočila

Oglejmo si Coppersmithov napad na RSA šifrirno funkcijo pri majhnem šifrirnem eksponentu, kjer pošiljamo sporočila, ki se delno ujemajo [9]. Naj bo sporočilo m sestavljeno iz dveh delov: znani del $B = 2^k b$, npr. 'Današnje geslo se glasi' in neznani del x, npr. 'qwert', katerega dolžina k je manjša kot $(\lg n)/e$.

TRDITEV 4.6 (Coppersmith) Naj bo (n, e) javni ključ. Naj za $m \in \mathbf{Z}_n^*$ velja $m = B + x$, kjer $B \in \mathbf{Z}_n^*$ in $|x| < n^{1/e}$. Tedaj lahko iz (B, n, e) in šifrirnega sporočila $c = m^e \bmod n$ učinkovito poiščemo m.

DOKAZ Šifrirano sporočilo je podano s $c = m^e = (B + x)^e \bmod n$. Če poznamo števila B, c, n in e, lahko uporabimo izrek 4.3 na polinomu $p(x) = (B + x)^e - c$ in učinkovito poiščemo x_0, ki zadošča

$$p(x_0) = (B + x_0)^e - c \equiv 0 \pmod{n},$$

če le obstaja tak x_0, za katerega velja $x_0 < n^{1/e}$. Potem je $m = B + x_0$. □

Poseben primer tega napada je pošiljanje kratkih sporočil m (primer $B = 0$), kajti če je $m < n^{1/e}$, potem je $m^e < n$. Tedaj lahko m izračunamo iz šifriranega sporočila $c = m^e \bmod n$ tako, da izračunamo e-ti koren števila c. V tem primeru si lahko pomagamo s soljenjem sporočila (§4.2 (i)).

Napad na stereotipna sporočila je primeren le za šifrirni eksponent $e = 3$. V tem primeru iščemo rešitve x_0, $|x_0| < n^{1/3}$, ki zadoščajo enačbi

$$p(x_0) = (B + x_0)^3 - c \equiv 0 \pmod{n},$$

Če je $B = 0$, velja $c = x_0^3$ v celih številih, saj je $x_0^3 < n$. S kubičnim korenjenjem dobimo $x_0 = c^{1/3}$.

Zgornja meja za x_0 je odvisna od modula n. Če ima x_0 npr. 250 bitov in modul n 512 bitov, potem z zgornjim napadom ne bomo našli x_0, saj je $x_0 > n^{1/3}$. Toda, če povečamo dolžino modula n na 1024 bitov, dolžino x_0 pa ohranimo na 250 bitih, so ti x_0 sedaj ranljivi na napad, ker je $x_0 < n^{1/3}$. Napad deluje tudi v primeru, ko x_0 predstavljajo biti z najvišjim redom [9]. V tem primeru množimo neznani del x sporočila m z 2^k.

(v) Napad delno znanega ključa

Naj bo (n, d) zasebni RSA ključ. Recimo, da je nasprotnik na nek način uspel izvedeti del dešifrirnega eksponenta d, npr. 1/4 bitov. Ali lahko izračuna ostale bite d? Odgovor je da, če je ustrezen šifrirni eksponent e majhen. Pred kratkim so Boneh, Durfee in Frankel [3] pokazali, da je mogoče izračunati d iz nekaj znanih bitov, če je $e < \sqrt{n}$. Rezultat nam pove, da je pomembno varovati zasebni RSA ključ.

TRDITEV 4.7 (Boneh, Durfee, Frankel) Naj bo (n, e) javni RSA ključ, kjer je modul n dolžine k bitov. Naj bo d ustrezen dešifrirni eksponent. Če poznamo $\lceil k/4 \rceil$ bitov dešifirnega eksponenta d z najnižjim redom, lahko rekonstruiramo cel d v času $O(e \cdot \lg e)$.

Dokaz trditve 4.7 temelji na še enem izreku Coppersmitha [9].

IZREK 4.8 (Coppersmith) Naj bo $n = pq$ RSA modul. Če poznamo $\lceil k/4 \rceil$ bitov števila p z najnižjim ali najvišjim redom, lahko učinkovito faktoriziramo n.

DOKAZ trditve 4.7. Označimo $\lceil k/4 \rceil$ bitov števila d z najnižjim redom z d_0. Torej $d_0 = d \bmod 2^{k/4}$. Po definiciji e in d obstaja tako število t, da je

$$ed - t\cdot\phi(n) = ed - t\cdot(n - p - q + 1) = 1.$$

Ker je $d < \phi(n)$, za t velja: $0 < t \leq e$. Enačbo reduciramo po modulu $2^{k/4}$ in vzamemo $q = n/p$. Sledi

$$ed_0 - t\cdot(n - p - n/p + 1) - 1 \equiv 0 \pmod{2^{k/4}}.$$

Ker poznamo $k/4$ bitov d z najnižjim redom, poznamo tudi $ed_0 \bmod 2^{k/4}$. Dobimo kvadratno enačbo

$$t\cdot p^2 - (ed_0 - t\cdot(n + 1) - 1)\, p + t\cdot n \equiv 0 \pmod{2^{k/4}}.$$

za spremenljivki t in p. Za vsako od e možnih vrednosti števila t, rešimo kvadratno enačbo za p in dobimo ustrezne vrednosti za $p \bmod 2^{k/4}$. Za vsako poženemo algoritem izreka 4.8 in poskusimo faktorizirati n. Pokažemo lahko [3], da je število primernih vrednosti za $p \bmod 2^{k/4}$ največ $e\cdot\lg e$. Zato po največ $e\cdot\lg e$ poskusih faktoriziramo n. Časovna zahtevnost predstavljenega algoritma je linearna v $e\cdot\lg e$ in polinomska v $\lg n$. □

Ker je časovna zahtevnost tega napada linearna v $e\cdot\lg e$, ga lahko izvedemo učinkovito le, če šifrirni eksponent e ni prevelik. Za šifrirni eksponent $e = 3$ napad izvedemo v sprejemljivem času. Za večje vrednosti, npr. $e = 2^{16} + 1 = 65537$, je napad sicer izvedljiv, vendar vzame precej več časa.

Vprašamo se lahko, ali lahko izvedemo enak napad pri majhnem šifrirnem eksponentu e, če poznamo bite d z najvišjim redom. Odgovor je ne. Razlog je v tem, da nam že majhen šifrirni eksponent da polovico bitov d z najvišjim redom. Ali drugače, polovico bitov d z najvišjim redom lahko dobimo že iz javnega ključa (e, n), če je e majhen. Zato nam poznavanje polovice bitov d ne razkrije preostalega d. To nam pove naslednja trditev.

TRDITEV 4.9 Recimo, da obstaja algoritem A, ki iz danih $k/2$ bitov dešifrirnega eksponenta d z navišjim redom izračuna cel d v času $T(k)$. Potem obstaja algoritem B, ki razbije RSA kriptosistem v času $e\cdot T(k)$.

DOKAZ Imamo enačbo $ed - t\cdot(n - p - q + 1) = 1$ za $0 < t \leq e$. Pri danem t lahko nasprotnik enostavno izračuna

$$\hat{d} = (t\cdot(n + 1) + 1)/e.$$

Potem je

$$|\hat{d} - d| = t(p + q)/e \leq t3\sqrt{n}\,/e < 3\sqrt{n}\,.$$

Zato je $\hat{d}$ dobra aproksimacija za d. Ocena pove, da je $k/2$ bitov $\hat{d}$ z najvišjim redom enakih tistim iz d. Zato, ko enkrat poznamo t, poznamo tudi $k/2$ bitov d z najvišjim redom. Algoritem B je potem tak: ker je le e možnih vrednosti za t, lahko nasprotnik za vsak t izračuna $\hat{d}$, požene algoritem A pri dani polovici bitov $\hat{d}$ z najvišjim redom. Ko najde pravi t, dobi tudi pravi d. □

4.3 Majhen dešifrirni eksponent

Podobno kot pri šifrirnem eksponentu *e*, nam izbira majhnega dešifrirnega eksponenta *d* omogoči učinkovitejše dešifriranje (v tem primeru izberemo najprej *d*, potem pa izračunamo *e*). Ker modularno eksponiranje poteka linearno v lg *d*, lahko majhen dešifrirni eksponent poveča hitrost dešifriranja vsaj za faktor 10 (pri 1024-bitnem modulu). Toda M. Wiener [25] je našel napad pri majhnem dešifrirnem eksponentu, ki povzroči popolni zlom RSA kriptosistema.

Dokaz trditve, ki opiše napad, temelji na aproksimaciji realnih števil z enostavnimi verižnimi ulomki. Ker verižni ulomki presegajo temo te diplomske naloge, ponovimo le za nas pomembna dejstva. Verižni ulomek

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{a_{k-1} + \cfrac{1}{a_k}}}}$$

pišemo $[a_0; a_1, \ldots, a_k]$, je *enostaven*, če je $a_0 \in \mathbf{Z}$, $a_i \in \mathbf{Z}$, $1 \le i \le k$, in velja $a_k \ge 2$, $1 \le k < \infty$. Za vsak verižni ulomek definiramo zaporedji $\{P_n\}$ in $\{Q_n\}$ z rekurzijo

$$P_{-2} = 0,\ P_{-1} = 1,\ P_n = a_n P_{n-1} + P_{n-2},$$
$$Q_{-2} = 1,\ Q_{-1} = 0,\ Q_n = a_n Q_{n-1} + Q_{n-2}.$$

Ulomek

$$\delta_n = P_n/Q_n$$

imenujemo *n-ta konvergenca* verižnega ulomka. Velja

$$\delta_i = [a_0; a_1, \ldots, a_i],\ 0 \le i \le k,$$

kar lahko hitro dokažemo z indukcijo na številu *i*. Vsako racionalno število *a*/*b* lahko zapišemo v končen enostavni verižni ulomek $[a_0; a_1, \ldots, a_k]$. Vrednosti a_i so enake kvocientom, ki jih dobimo na posameznem koraku Evklidovega algoritma pri podatkih *a* in *b*. Vsako realno število *r* lahko zapišemo v verižni ulomek po *metodi verižnih ulomkov* takole:

$$\alpha_0 = r \text{ in } a_i = \lfloor \alpha_i \rfloor,\ \alpha_{i+1} = \frac{1}{\alpha_i - a_i},\ \text{če } a_i \ne \alpha_i,\ \text{za } i = 0, 1, 2, \ldots.$$

Verižni ulomki so primerni za aproksimacijo realnih števil. Med drugimi velja naslednji izrek:

IZREK 4.10 Če sta $a, b \in \mathbf{Z}$, $b \ge 1$, *r* realno število in velja $|r - a/b| \le 1/2b^2$, potem je ulomek *a*/*b* enak neki konvergenci enostavnega verižnega ulomeka števila *r*.

Dokaz najdemo v [12, str 153]. Vrnimo se sedaj k Winerjevemu napadu na RSA kriptosistem pri majhnem dešifrirnem eksponentu.

TRDITEV 4.11 (Wiener) Naj bo $n = pq$, kjer $q < p < 2q$. Naj bo $d \le 1/3 \cdot n^{1/4}$. Potem lahko nasprotnik iz javnega ključa (n, e) učinkovito izračuna dešifrirni eksponent d.

DOKAZ Ker je $ed \equiv 1 \pmod{\phi(n)}$, obstaja tako pozitivno celo število k, da je $ed - 1 = k\phi(n)$. Če enačbo delimo z $d\phi(n)$ dobimo

$$\frac{e}{\phi(n)} - \frac{k}{d} = \frac{1}{d\phi(n)},$$

kar pomeni, da je ulomek $e/\phi(n)$ aproksimacija ulomeka k/d. Ker nasprotnik ne pozna vrednosti $\phi(n)$ jo lahko aproksimira z modulom n. Res, ker je $\phi(n) = n - (p + q) + 1$ in velja $p + q - 1 < 3\sqrt{n}$, velja ocena $|n - \phi(n)| \le 3\sqrt{n}$. Zato je

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{ed - k\phi(n) - kn + k\phi(n)}{nd}\right| = \left|\frac{1 - k(n - \phi(n))}{nd}\right| < \frac{3k\sqrt{n}}{nd} = \frac{3k}{d\sqrt{n}}.$$

Velja $k\phi(n) = ed - 1 < ed$. Ker je $e < \phi(n)$, je $k < d \le 1/3 \cdot n^{1/4}$. Tako dobimo

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{d \cdot n^{1/4}} < \frac{1}{2d^2}.$$

Po izreku 4.10 je ulomek k/d ena od konvergenc enostavnega verižnega ulomka $[a_0; a_1, \ldots, a_l]$ za ulomek e/n. Verižni ulomek $[a_0; a_1, \ldots, a_l]$ izračunamo v času $O((\lg n)^2)$ s pomočjo Evklidovega algoritma. Vse, kar je potrebno storiti, je izračunati l konvergenc verižnega ulomka za e/n, kjer je število l po posledici 5.3 omejeno z $O(\lg n)$. Ulomek k/d bo enak eni od konvergenc $\delta_i = P_i/Q_i$, $1 \le i \le l - 1$. Izračun P_i in Q_i naredimo v času $O((\lg n)^2)$. Ker je $ed - k\phi(n) = 1$, velja $D(k, d) = 1$, in zato je k/d okrajšan ulomek. Pri vsakem kandidatu za k/d pogledamo ali je $\phi(n) = (ed - 1)/k$ celo število, potem pa poskusimo faktorizirati n tako kot v §4.1 (iii). To je polinomski algoritem za izračun dešifrirnega eksponenta. □

Ker je n ponavadi 1024 bitno število, mora biti d vsaj 256 bitno število, da se izognemo temu napadu. To v primeru pametnih kartic ni najbolj vzpodbudna novica, saj manjši dešifrirni eksponent pomeni večji prihranek energije. Oglejmo si dva predloga, ki omogočata hitrejše dešifriranje in preprečita ta napad.

Prvi je velik šifrirni eksponent. Recimo, da namesto šifrirnega eksponenta e izberemo e', kjer je $e' = e + t\phi(n)$ za nek velik t. Potem bo d še vedno isti, število k iz enačbe $ed - 1 = k\phi(n)$ pa ne bo več majhno in zgornji napad ne pride več v poštev. Če je $e' > n^{3/2}$, potem ni važno kako majhen je dešifrirni eksponent d. Ker je $e'/n > n^{1/2}$, je $k/d > n^{1/2}$. Zato oceno $3k/dn^{1/2}$ ne moremo omejiti pod $1/2d^2$ za $d > 1$. Napada ne moremo izvesti.

Drugi predlog je Kitajski izrek o ostankih (izrek 2.11). Recimo, da izberemo tak d, da sta števili

$$d_p = d \bmod (p - 1) \text{ in } d_q = d \bmod (q - 1)$$

majhni, npr. 128 bitov. Potem lahko približno štirikrat (glej §5.5) hitreje dešifriramo šifrirano sporočilo c na naslednji način. Najprej izračunamo

$$m_p = c^{d_p} \bmod p \text{ in } m_q = c^{d_q} \bmod q.$$

Uporabimo Kitajski izrek o ostankih, da dobimo sporočilo $m \in \mathbf{Z}_n$, ki zadošča kongruencama $m = m_p \bmod p$ in $m = m_q \bmod q$. Zanj velja $m = c^d \bmod n$. Čeprav sta števili d_p in d_q majhni, zgornji napad v tem primeru ne bo deloval, saj je vrednost d velika, tj. reda velikosti $\phi(n)$.

4.4 Napadi pri nepravilni uporabi

V tem razdelku bomo predstavili nekaj napadov na RSA kriptosistem, ki sami po sebi ne predstavljajo kakšne posebne nevarnosti za RSA kriptosistem, pač pa predstavljajo nevarnost le v primeru nepravilne ali neprevidne uporabe RSA kriptosistema.

(i) Napad z enostavnim iskanjem

Če je sporočilo m majhno (npr. do 32 bitov pri 1024-bitnem modulu n) ali predvidljivo, potem lahko nasprotnik odšifrira šifrirano sporočilo c tako, da enostavno zašifrira vse možna sporočila take dolžine, dokler ne dobi c. Soljenje sporočila, kot je opisano v §4.2 (i), je enostaven način za preprečitev tega napada.

(ii) Multiplikativna lastnost

RSA šifrirna in dešifrirna funkcija imata naslednjo multiplikativno lastnost. Naj bosta m_1 in m_2 dve sporočili ter c_1 in c_2 ustrezni šifrirni sporočili. Velja

$$(m_1 \cdot m_2)^e \equiv m_1^e \cdot m_2^e \equiv c_1 \cdot c_2 \pmod{n}.$$

Z drugimi besedami, šifrirano sporočilo, ki ustreza sporočilu $m = m_1 \cdot m_2 \bmod n$, je $c = c_1 \cdot c_2 \bmod n$.

Multiplikativna lastnost vodi k naslednjemu napadu na RSA šifrirno funkcijo. Recimo, da želi nasprotnik dešifrirati šifrirano sporočilo $c = m^e \bmod n$ namenjeno osebi A. Predpostavimo, da oseba A dešifrira nasprotniku neko drugo šifrirano sporočilo. Nasprotnik lahko skrije c z izbranim številom $x \in \mathbf{Z}_n^*$ in izračuna

$$\bar{c} = cx^e \bmod n.$$

Ko bo oseba A dobila $\bar{c}$, bo za nasprotnika izračunala $\bar{m} = (\bar{c})^d \bmod n$. Ker je

$$\bar{m} \equiv (\bar{c})^d \equiv c^d (x^e)^d \equiv mx \pmod{n},$$

lahko nasprotnik izračuna $m = \bar{m}\, x^{-1} \bmod n$.

Temu napadu se lahko v praksi izognemo tako, da vpeljemo strukturne omejitve sporočil. Če se šifrirano sporočilo c odšifrira v sporočilo, ki nima določene strukture, potem je c zavržen kot goljufiv. Tako napad ne bo uspel, saj oseba A ne bo odšifrirala $\bar{c}$ za nasprotnika. Podoben napad lahko izvedemo tudi pri RSA elektronskem podpisu.

(iii) Napad istega modula

Poglejmo, zakaj je važno, da ima vsaka oseba različen RSA modul n. Včasih se je predlagalo, da naj bi avtoriteta, ki ji vsi zaupajo, izbrala en sam RSA modul n in potem razdelila različne šifrirne/dešifrirne eksponente (e_i, d_i) vsaki osebi posebej. Toda, kot smo pokazali v §4.1 (ii), lahko s poznavanjem kateregakoli para (e_i, d_i) faktoriziramo modul n in tako lahko katerakoli oseba izve dešifrirne eksponente vseh ostalih.

(iv) Ciklični napadi

Naj bo $c = m^e \bmod n$ šifrirano sporočilo. Definirajmo zaporedje

$$x_0 = c,\ \ x_i = (x_{i-1})^e \bmod n.$$

Ker je RSA šifrirna funkcija injektivna in velja $x_i \in \{0, 1, \ldots, n-1\}$ za vsak i, se bo v tem zaporedju prvi ponovil x_0. Naj bo k tako pozitivno število, da je $x_0 = x_k$. Velja

$$c^{e^k} \equiv c \pmod{n}.$$

Od tod sledi $c^{e^{k-1}} \equiv m \pmod{n}$, kar nas pripelje do *cikličnega napada* na RSA kriptosistem. Nasprotnik izračuna $c^e \bmod n$, $c^{e^2} \bmod n$, $c^{e^3} \bmod n$, …, dokler prvič ne dobi c. Če za število k velja $c^{e^k} \bmod n = c$, potem je prejšnje število v ciklu, $c^{e^{k-1}} \bmod n$, enako m.

Posplošen ciklični napad je napad, pri katerem poiščemo najmanjše pozitivno število u, tako da je $f = D(c^{e^u} - c, n) > 1$. Če je

$$c^{e^u} \equiv c \pmod{p} \ \text{ in } \ c^{e^u} \not\equiv c \pmod{q},$$

potem je $f = p$. Podobno, če je

$$c^{e^u} \not\equiv c \pmod{p} \ \text{ in } \ c^{e^u} \equiv c \pmod{q},$$

potem je $f = q$. V obeh primerih faktoriziramo modul n, potem pa lahko izračunamo dešifrirni eksponent d in nazadnje sporočilo m. Po drugi strani, če velja

$$c^{e^u} \equiv c \pmod{p} \ \text{ in } \ c^{e^u} \equiv c \pmod{q},$$

potem je $f = n$ in $c^{e^u} \equiv c \pmod{n}$. V resnici mora biti u najmanjše pozitivno število k, za katerega $c^{e^k} \equiv c \pmod{n}$. V tem primeru je ciklični napad uspel in tako lahko učinkovito izračunamo $m = c^{e^{u-1}} \bmod n$. Ker se ta primer zgodi manj pogosto kot prva dva, se posplošen ciklični napad ponavadi konča preden se konča ciklični napad. Zaradi tega razloga lahko gledamo na posplošen ciklični napad kot na algoritem za faktorizacijo n. Ker za faktorizacijo n predpostavljamo, da je težak problem, ti ciklični napadi ne predstavljajo grožnjo za varnost RSA kriptosistema.

(v) Odkrita sporočila

Sporočilo m, $0 \le m \le n-1$, imenujemo *odkrito*, če se zašifrira samo vase, tj. $m^e = m \bmod n$. Vedno obstaja nekaj sporočil, ki so odkrita (npr. $m = 0$, $m = 1$ in $m = n-1$).

TRDITEV 4.12 Število odkritih sporočil pri RSA je enako $[1+D(e-1, p-1)]\cdot[1+D(e-1, q-1)]$.

DOKAZ Iščemo število takih sporočil m, za katere velja $m^e = m \bmod n$. Ker je $n = pq$, kjer sta p in q praštevili, je dovolj gledati kongruenci $m^e \equiv m \pmod{p}$ in $m^e \equiv m \pmod{q}$. Recimo, da je $m \in \mathbf{Z}_p^*$. Potem dobimo $m^{e-1} \equiv 1 \pmod{p}$. Ker je p praštevilo, je po izreku 2.20 multiplikativna grupa $\mathbf{Z}_p^*$ ciklična: obstaja element $\alpha \in \mathbf{Z}_p^*$, ki ima red $p-1$ in velja $\mathbf{Z}_p^* = \{\alpha^i \mid 0 \le i \le p-2\}$. Torej lahko zapišemo $m = \alpha^j \bmod p$. Sledi $\alpha^{j(e-1)} \equiv 1 \pmod{p}$. Potem red elementa α po trditvi 2.15 deli $i(e-1)$.

Torej velja enakost

$$i(e-1) = k(p-1),$$

za neko število $k \in \mathbf{Z}$. Zanima nas, koliko je takih števil i. Zapišimo

$$(e-1),\ 2(e-1),\ 3(e-1),\ \ldots,\ (p-1)(e-1).$$

Najmanjši i, pri katerem velja zgornja enačba, je najmanjši skupni večkratnik števil $p-1$ in $e-1$. Ker velja $v(p-1,\ e-1)\cdot D(p-1,\ e-1) = (p-1)(e-1)$, je število rešitev m, ki zadoščajo kongruenci $m^{e-1} \equiv 1 \pmod{p}$ enako $D(p-1,\ e-1)$. Če upoštevamo še rešitev $m = 0$, je v $\mathbf{Z}_p$ število rešitev enako $1 + D(p-1,\ e-1)$. Podobno obravnavamo primer $m^e \equiv m \pmod{q}$. Potem pa s Kitajskim izrekom o ostankih (izrek 2.11) rešitve še združimo in dobimo željeno število rešitev. □

Ker so $e-1$, $p-1$ in $q-1$ soda števila, je število odkritih sporočil vedno vsaj 9. Če sta p in q naključni praštevili in e naključno izbran (ali pa majhen, npr. $e = 3$ ali $e = 2^{16} + 1 = 65537$), potem je delež sporočil, ki so odkrita z RSA šifriranjem, v splošnem zanemarljivo majhen in zato odkrita sporočila v praksi ne predstavljajo grožnje k varnosti RSA šifriranja.

(vi) Prikrivanje sporočil

Naj ima podpisnik zasebni ključ (n, d) in ustrezni javni ključ (n, e). Recimo, da želi nasprotnik pridobiti podpis za sporočilo $m \in \mathbf{Z}_n^*$. Seveda mu ga podpisnik ne bo dal kar tako. Zato naredi naslednje. Vzame naključen $r \in \mathbf{Z}_n^*$ in izračuna

$$m' = r^e m \bmod n.$$

Potem prosi podpisnika naj mu podpiše naključno sporočilo m'. Podpisnik bo morda sedaj, nič hudega sluteč, podpisal m'.

Toda poglejmo, kaj se v resnici zgodi. Naj bo s' podpis za m'. Potem velja $s' = (m')^d \bmod n$. Nasprotnik sedaj enostavno izračuna $s = s'/r$ in pridobi podpis za m, saj velja

$$s^e = (s')^e/r^e = (m')^{ed}/r^e \equiv m'/r^e = m \pmod{n}.$$

Ta napad nasprotniku omogoča pridobiti veljaven podpis za sporočilo po njegovi izbiri, tako da prosi podpisnika, da mu podpiše naključno prikrito sporočilo. Ker v praksi večina podpisnih shem uporablja zgoščevalne funkcije pred podpisovanjem, ta napad ne predstavlja večje skrbi.

4.5 Implementacijski napadi

Na koncu poglavja si na kratko oglejmo čisto drugačen razred napadov. Namesto, da bi napadla RSA šifrirno funkcijo, sta naslednja napada usmerjena na določen tip implementacije RSA kriptosistema.

(i) Časovni napad

Recimo, da smo prišli do pametne kartice, cf. [15], npr. zdravstvene kartice, ki ima shranjen zasebni RSA ključ. Čeprav iz pametne kartice ne moremo kar tako prebrati zasebni ključ, je Kocher [18] pokazal, da je z natančnim merjenjem časa, ki ga potrebuje pametna kartica za RSA dešifriranje (ali podpis), mogoče presenetljivo hitro priti do zasebnega dešifrirnega eksponenta.

Kocherjevo idejo bomo opisali le v grobem. Oglejmo si enostavno implementacijo RSA kriptosistema z algoritmom kvadriraj-in-množi (glej §5.4, algoritem 16). Naj bo dešifrirni eksponent $d = d_n d_{n-1} \ldots d_0$ v binarni reprezentaciji, tj. $d_i \in \{0,1\}$. Algoritem kvadriraj-in-množi izračuna podpis

$$c = m^d \bmod n = \prod_{i=0}^{k} m^{2^i d_i} \bmod n = (m^{2^0})^{d_0} (m^{2^1})^{d_1} \ldots (m^{2^t})^{d_n} \bmod n$$

z $n-1$ modularnimi kvadriranji in $\sum_{i=0}^{n} d_i - 1$ modularnimi množenji, glej §5.4 (iv).

Napad začnemo tako, da s pametno kartico podpišemo večje število naključno generiranih sporočil $m_1, \ldots, m_k \in \mathbf{Z}_n^*$ in izmerimo čase T_i, ki so potrebni za izračun podpisa

$$c_i = m_i^d \bmod n.$$

Napad odkriva posamezne bite d. Začne se z bitom d_0. Ker je d multiplikativni inverz šifrirnega eksponenta e po modulu $\phi(n)$, je $D(d, \phi(n)) = 1$. Ker je $\phi(n)$ sodo število, mora biti d liho število. Zato je bit $d_0 = 1$. Sledi bit d_1. Če je bit $d_1 = 1$, potem pametna kartica izračuna $m^2 \bmod n$ in $m \cdot m^2 \bmod n$, sicer samo $m^2 \bmod n$. Naj bodo t_i časi, ki jo potrebuje pametna kartica, da izračuna vrednosti

$$m_i \cdot m_i^2 \bmod n$$

za vsako sporočilo m_i posebej. Čase t_i izmerimo pred napadom. Kocher je opazil, da so časi $\{t_i\}$ in $\{T_i\}$ med seboj povezani [18]. Če nasprotnik izmeri to povezavo, potem lahko ugotovi ali je d_1 enak 1 ali 0. Podobno lahko ugotovi bite d_2, d_3, itd. Če je šifrirni eksponent e majhen, potem lahko z delnim poznavanjem d uporabi napad delno znanega ključa (§4.2 (v)).

Najpreprostejša obramba pred tem napadom je, da poskrbimo, da se modularno eksponiranje vedno izvede v konstantnem času, npr. na vsakem koraku algoritma kvadriraj-in-množi dodamo ustrezen časovni zamik, preden izvedemo nov korak. Vendar so take implementacije RSA kriptosistema počasne. Drug pristop je prikrivanje (glej §4.4 (vi)). Pametna kartica izbere naključno število $r \in \mathbf{Z}_n^*$ in izračuna $m' = m \cdot r^e \bmod n$. Potem uporabi d in izračuna $c' = (m')^d \bmod n$. Končno določi $c = c'/r \bmod n$. Na ta način pametna kartica izračuna podpis na naključnem sporočilu, ki ga napadalec ne pozna.

Pred kratkim je Kocher predstavil podoben napad. Pokazal je, da je z natančno meritvijo porabe toka pri pametnih karticah mogoče odkriti dešifrirni eksponent d. Kot se je izkazalo, porabi pametna kartica pri modularnem množenju več energije kot sicer. Z merjenjem časa povečane porabe toka, lahko napadalec ugotovi ali je kartica izvedla eno ali dve modularni operaciji in tako razkrije bite dešifrirnega eksponenta d.

Na koncu se lahko še vprašamo ali je naša nova zdravstvena kartica varna pred tem napadom.

(ii) Naključne napake

Boneh, DeMillo in Lipton [4] so opazili, da obstaja nevarnost pri uporabi Kitajskega izreka o ostankih (glej §5.5). Recimo, da med nastajanjem podpisa pride do kratkega stika v podpisnikovem računalniku in povzroči en napačen izračun. Npr. pri prepisovanju podatkov v registru z laserjem povzročimo spremembo nekega bita. Potem lahko s poznavanjem nepravilnega podpisa znanega sporočila faktoriziramo podpisnikov modul n.

Recimo, da se eno od števil $c_p = m^{d_p} \bmod p$ in $c_q = m^{d_q} \bmod q$, kjer je $d_p = d \bmod (p-1)$ in $d_q = d \bmod (q-1)$, ne izračuna pravilno. Naj se c_p izračuna pravilno, $\hat{c}_q$ pa ne. Podpis bo tako enak

$$\hat{c} = T_1 c_p + T_2 \hat{c}_q ,$$

kjer sta T_1 in T_2 koeficienta rešitve Kitajskega izreka o ostankih. Ko dobimo in preverimo $\hat{c}$, vemo, da je podpis napačen, saj $\hat{c}^e \not\equiv m \pmod{n}$. Toda velja

$$\hat{c}^e \equiv m \pmod{p} \text{ in } \hat{c}^e \not\equiv m \pmod{q}.$$

Potem je $D(n, \hat{c}^e - m) = p$. Če poleg nepravilnega podpisa $\hat{c}$ za sporočilo m poznamo še pravilen podpis c, lahko dobimo netrivialen faktor modula n tudi tako, da izračunamo $D(n, \hat{c} - c)$.

Naključne napake grozijo številnim kriptosistemom, tudi implementacijam RSA kriptosistema, ki ne uporabljajo Kitajskega izreka o ostankih [4]. Enostavna obramba pred tem napadom, je da vedno preverimo podpis, preden ga pošljemo. To ne zahteva veliko časa, če je šifrirni eksponent e majhen. Napad lahko preprečimo tudi, če uporabimo postopek soljenja sporočil (glej §4.2 (i)).

5. IMPLEMENTACIJA RSA

RSA kriptosistem uporablja zelo velika števila (npr. 1024-bitna števila ≈ 309-mestna desetiška števila) za doseganje nivoja varnosti, ki ga potrebujejo današnje aplikacije. Največji izziv pri implementaciji RSA kriptosistema predstavlja vpeljava velikih števil in z njimi povezane osnovne operacije in algoritmi. Zato bomo v prvi polovici tega poglavja navedli algoritme za učinkovito računanje z velikimi števili, torej operacije s celimi števili in števili ostankov po modulu n, in obravnavali njihovo časovno zahtevnost. Izračun dešifrirnega eksponenta zahteva poznavanje razširjenega Evklidovega algoritma. Opisali bomo tako Evklidov algoritem kot tudi njegovo razširjeno obliko. Za implementacijo RSA kriptosistema potrebujemo velika praštevila. Pri iskanju praštevil si bomo pomagali z Miller-Rabinovim praštevilskim testom. Za implementacijo sheme elektronskega podpisa IFSSA po standardu IEEE P1363 [14] potrebujemo zgoščevalno funkcijo SHA-1. Podali bomo tako algoritem za SHA-1 kot tudi shemo elektronskega podpisa IFSSA. V zaključku poglavja bomo na kratko predstavili še program "RSA Encrypter&Signer", ki je del diplomske naloge in je priložen na CD-ROM-u. Viri: [1], [19] in [22].

5.1 Reprezentacija števil

Pozitivna cela števila lahko predstavimo na več načinov. Najbolj pogosta je reprezentacija števil pri osnovi 10. Npr. $A = 123$ pri osnovo 10 pomeni $A = 1\cdot10^2 + 2\cdot10^1 + 3\cdot10^0$. Pri računanju z računalniki je bolj primerna osnova 2, t.i. *binarna* reprezentacija.

Če je $b \geq 2$ celo število, potem lahko vsako pozitivno celo število a enolično predstavimo kot

$$a = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0,$$

kjer $0 \leq a_i \leq b$ za $0 \leq i \leq n$ in $a_n \neq 0$. Predstavitev pozitivnega števila a kot vsote potenc števila b imenujemo *reprezentacija števila a pri osnovi b* (oziroma *b-reprezentacija števila a*), pišemo $a = (a_n a_{n-1} \ldots a_1 a_0)_b$. Števila a_i, $0 \leq i \leq n$, imenujemo *cifre* ali *števke*, a_n je cifra z *najvišjim redom*, a_0 pa cifra z *najnižjim redom*. Če je $b = 10$, pišemo $a = a_n a_{n-1} \ldots a_1 a_0$. *Dolžina* reprezentacije števila $a = (a_n a_{n-1} \ldots a_1 a_0)_b$ pri osnovi b je $n + 1$. Večkrat je priročno dodati ničle k cifri najvišjega reda in s tem podaljšati reprezentacijo na izbrano dolžino.

ALGORITEM 6 - Reprezentacija pri osnovi b

Podatki: celi števili a in b, kjer $a \geq 0$ in $b \geq 2$.
Rezultat: reprezentacija števila a pri osnovi b, kjer $n \geq 0$ in $a_n \neq 0$, če $n \geq 1$.

1. $i := 0,\ x := a,\ q := \lfloor x/b \rfloor,\ a_i := x - qb.$
2. while $(q > 0)$ do
   2.1 $i := i + 1,\ x := q,\ q := \lfloor x/b \rfloor,\ a_i := x - qb.$
3. return $((a_n a_{n-1} \ldots a_1 a_0)_b)$;

Negativna števila lahko predstavimo na več načinov. Dva pogosta načina sta:

(i) Predznačena reprezentacija. Predznak števila in njegova absolutna vrednost sta predstavljena ločeno v predznačeni reprezentaciji. Ponavadi je pozitivno število določeno s predznačeno cifro 0, negativno število pa s predznačeno cifro $b - 1$. Za reprezentacijo števil pri osnovi b dolžine n je od b^n možnih števil samo $2b^{n-1}$ uporabljenih. Bolj natančno $b^{n-1} - 1$ je pozitivnih, natančno $b^{n-1} - 1$ je negativnih, 0 pa ima dve reprezentaciji. Tabela 2 predstavlja binarno predznačeno reprezentacijo števil $[-7, 7]$.

Slaba lastnost predznačene reprezentacije je ta, da moramo pri določenih operacijah, kot sta seštevanje ali odštevanje, preveriti predznačeno cifro, če želimo ugotoviti s kakšnim številom imamo opravka. To je lahko časovno zahtevno, če imamo veliko operacij.

(ii) Komplementarna reprezentacija. Seštevanje in odštevanje pri komplementarni reprezentaciji števil ne zahteva preverjanje predznačene cifre. Nenegativna števila v razponu $[0, b^{n-1} - 1]$ so predstavljena z zaporedjem cifer pri osnovi b dolžine n s cifro z najvišjim redom 0. Če je pozitivno število x v tem razponu predstavljeno kot $(x_n x_{n-1} \ldots x_1 x_0)_b$, kjer je $x_n = 0$, potem je njegova negativa reprezentacija $-x$, predstavljena kot $\overline{x} = (\overline{x}_n \overline{x}_{n-1} \ldots \overline{x}_1 \overline{x}_0)_b + 1$, kjer $\overline{x}_i = b - 1 - x_i$ in + standardno seštevanje s prenosom. Tabela 2 predstavlja binarno komplementarno reprezentacijo števil [–7, 7].

Zaporedje	Predznačena	Komplementarna
0111	7	7
0110	6	6
0101	5	5
0100	4	4
0011	3	3
0010	2	2
0001	1	1
0000	0	0

Zaporedje	Predznačena	Komplementarna
1111	–7	–1
1110	–6	–2
1101	–5	–3
1100	–4	–4
1011	–3	–5
1010	–2	–6
1001	–1	–7
1000	–0	–8

TABELA 2: Predznačena in komplementarna reprezentacija števil [–7, 7].

5.2 Osnovne operacije v Z

Naj bosta a in b nenegativni celi števili, obe manjši ali enaki številu n. Število bitov binarne reprezentacije števila n je $\lg n = \lfloor \log_2 n \rfloor + 1$, ki ga lahko aproksimiramo z $\log_2 n$. V tabeli 3 je predstavljena časovna zahtevnost štirih osnovnih operacij, seštevanja, odštevanja, množenja in deljenja z uporabo klasičnih algoritmov, ki jih bomo predstavili v nadaljevanju. Algoritmi, ki opišejo te operacije, predpostavljajo uporabo predznačene reprezentacije števil, kjer je predznak impliciten. Časovna zahtevnost operacije je enaka oceni za število enostavnih operacij, ki jih pri danem algoritmu izvede procesor (glej §2.3).

Operacija		Zahtevnost
Seštevanje	$a + b$	$O(\lg a + \lg b) = O(\lg n)$
Odštevanje	$a - b$	$O(\lg a + \lg b) = O(\lg n)$
Množenje	$a \cdot b$	$O((\lg a)(\lg b)) = O((\lg n)^2)$
Deljenje	$a = q \cdot b + r$	$O((\lg q)(\lg b)) = O((\lg n)^2)$

TABELA 3: Računska zahtevnost osnovnih operacij v **Z**.

(i) Seštevanje in odštevanje

Seštevanje in odštevanje se izvaja na dveh celih številih enake dolžine pri osnovi b. Če hočemo sešteti ali odšteti dve različno dolgi števili, moramo krajšemu dodati ustrezno število ničel z leve (tj. cifri z najvišjim redom).

ALGORITEM 7 - Natančno seštevanje

Podatki: dve pozitivni celi števili x in y, obe dolžine $n + 1$ pri osnovi b.
Rezultat: vsota $x + y = (w_n w_{n-1} \ldots w_1 w_0)_b$.

1. $c := 0$; (c je prenosna cifra)
2. for $i := 0$ to n do
   2.1 $w_i := (x_i + y_i + c) \bmod b$;
   2.2 if $(x_i + y_i + c < b)$ then $c := 0$ else $c := 1$;
3. $w_{n+1} := c$;
4. return $(w_n w_{n-1} \ldots w_1 w_0)_b$;

ALGORITEM 8 - Natančno odštevanje

Podatki: dve pozitivni celi števili x in y , obe dolžine $n + 1$ pri osnovi b, kjer $x \geq y$.
Rezultat: razlika $x - y = (w_n w_{n-1} \ldots w_1 w_0)_b$.

1. $c := 0$; (c je prenosna cifra)
2. for $i := 0$ to n do
   2.1 $w_i := x_i - y_i + c$;
   2.2 if $(x_i - y_i + c \geq 0)$ then $c := 0$ else $c := -1$;
3. return $(w_n w_{n-1} \ldots w_1 w_0)_b$;

Opomba: Pogoj $x \geq y$ lahko tudi odstranimo. Če absolutne vrednosti x in y niso znane, naredimo naslednje. Če je $c = -1$ pri zaključku algoritma 8, potem ponovimo algoritem 8 z $x = (00\ldots00)_b$ in $y = (w_n w_{n-1} \ldots w_1 w_0)_b$. Pregledu absolutnih vrednosti x in y se lahko izognemo tudi z uporabo komplementarne reprezentacije.

Če je osnova b izbrana tako, da lahko korak 2.1 algoritma 7 izračunamo s strojno opremo, potem algoritem 7 zahteva največ $2(n + 1)$ enostavnih seštevanj. Podobno zahteva algoritem 8 največ $2(n + 1)$ enostavnih odštevanj.

(ii) Množenje

Naj bosta x in y celi števili izraženi v b-reprezentaciji: $x = (x_n x_{n-1} \ldots x_1 x_0)_b$ in $y = (y_t y_{t-1} \ldots y_1 y_0)_b$. Produkt $x \cdot y$ bo imel največ $(n + t + 2)$ cifer pri osnovi b. Algoritem 9 je v bistvu standardna metoda množenja, ki se jo učimo v osnovni šoli. Pri tem predpostavljamo, da se cifre med seboj množijo z enostavnim množenjem, ki ga zmore procesor. Učinkovita implementacija te operacije v procesorju je ključnega pomena za učinkovito implementacijo algoritma 9.

ALGORITEM 9 - Natančno množenje

Podatki: pozitivni celi števili x in y, dolžine $n + 1$ in $t + 1$ pri osnovi b.
Rezultat: produkt $x \cdot y = (w_n w_{n-1} \ldots w_1 w_0)_b$.

1. for $i := 0$ to $(n + t + 1)$ do $w_i := 0$; (inicializacija)
2. for $i := 0$ to t do
   2.1 $c := 0$;
   2.2 for $j := 0$ to n do $(uv)_b := w_{i+j} + x_j \cdot y_i + c$, $w_{i+j} := v$, $c := u$;
   2.3 $w_{i+n+1} := u$;
3. return $(w_n w_{n-1} \ldots w_1 w_0)_b$;

Računsko najbolj zahteven del algoritma 9 je korak 2.2. Izračun $w_{i+j} + x_j \cdot y_i + c$ imenujemo *notranje množenje*. Ker so w_{i+j}, x_j, y_i in c cifre pri osnovi b, je rezultat notranjega množenja največ

$$(b-1) + (b-1)^2 + (b-1) = b^2 - 1.$$

Zato lahko rezultat predstavimo kot dve cifri pri osnovi b, $(uv)_b$, kjer sta u in v cifri pri osnovi b, u pa je lahko tudi 0. Algoritem 9 zahteva $(n+1)(t+1)$ enostavnih množenj.

(iii) Kvadriranje

Tudi pri kvadriranju bomo imeli oznako $(uv)_b$, kjer dovoljujemo, da lahko vrednost u predstavlja dve cifri pri osnovi b, vrednost v pa je vedno ena cifra pri osnovi b.

ALGORITEM 10 - Natančno kvadriranje

Podatki: pozitivno celo število $x = (x_t x_{t-1} \ldots x_1 x_0)_b$.
Rezultat: produkt $x \cdot x = x^2 = (w_{2t-1} w_{2\,t-2} \ldots w_1 w_0)_b$.

1. for $i := 0$ to $(2t-1)$ do $w_i := 0$; (inicializacija)
2. for $i := 0$ to $(t-1)$ do
   2.1 $(uv)_b := w_{2i} + x_i \cdot x_i$, $w_{2i} := v$, $c := u$;
   2.2 for $j := (i+1)$ to $(t-1)$ do $(uv)_b := w_{i+j} + 2x_j \cdot x_i + c$, $w_{i+j} := v$, $c := u$;
   2.3 $w_{i+t} := u$;
3. return $(w_{2t-1} w_{2t-2} \ldots w_1 w_0)_b$;

Opomba: V koraku 2.2 je lahko vrednost u večja od ene cifre, za kar moramo poskrbeti. Pokažimo, da velja $0 \le u \le 2(b-1)$. Na začetku so v koraku 2.2 vse cifre $\le b-1$, zato velja

$$w_{i+j} + 2x_j \cdot x_i + c \le (b-1) + 2(b-1)^2 + (b-1) = 2b(b-1).$$

To pomeni, da je vrednost $u \le 2(b-1)$. Ker w_{i+j} vedno postane v, je $w_{i+j} \le b-1$. Ker c vedno postane u, je $c \le 2(b-1)$. Zato v vseh nadaljnih iteracijah koraka 2.2 velja ocena

$$w_{i+j} + 2x_j \cdot x_i + c \le (b-1) + 2(b-1)^2 + 2(b-1) = 2b(b-1) + b - 1,$$

iz česar sledi $u \le 2(b-1)$ in $v \le b-1$. Podobno se lahko prepričamo, da je c v koraku 2.1 manjši od $b-1$, tudi če je kak $w_{2i} \le 2(b-1)$.

Računsko zahteven del algoritma 10 je korak 2. Enostavnih množenj je približno $(t^2 + t)/2$, če ne upoštevamo množenje z 2, kar procesor izvede z ukazom SHL (pomik v levo). To je približno polovica manj enostavnih množenj kot v algoritmu 9. Kvadriranje lahko uporabimo tudi pri operaciji množenja. Ker velja identiteta

$$x \cdot y = ((x+y)^2 - (x-y)^2)/4,$$

lahko produkt $x \cdot y$ izračunamo z dvema kvadriranjema. Povečanje hitrosti za faktor 2 pa se že pozna pri aplikacijah, v katerih veliko množimo (npr. pri aplikacijah RSA kriptosistema).

(iv) Deljenje

Deljenje je najbolj zapletena in časovno zahtevna osnovna računska operacija, zato lahko s premišljeno implementacijo deljenja z velikimi števili pridobimo največ računskega časa. Algoritem 11 izračuna kvocient q in ostanek r v b-reprezentaciji, ko število x deli s številom y.

ALGORITEM 11 - Natančno deljenje

Podatki: pozitivni celi števili $x = (x_n x_{n-1} \ldots x_1 x_0)_b$ in $y = (y_t y_{t-1} \ldots y_1 y_0)_b$, $n \geq t \geq 1$, $y_t \neq 0$.
Rezultat: kvocient $q = (q_{n-t}\, q_{n-t-1} \ldots q_1 q_0)_b$ in ostanek $r = (r_t\, r_{t-1} \ldots r_1 r_0)_b$, kjer $x = qy + r$, $0 \leq r \leq y$.

1. for $j := 0$ to $(n - t)$ do $q_j := 0$; (inicializacija)
2. while $(x \geq yb^{n-t})$ do $q_{n-t} := q_{n-t} + 1$, $x := x - yb^{n-t}$;
3. for $i := n$ downto $(t + 1)$ do
   3.1 if $(x_i = y_t)$ then $q_{i-t-1} := b - 1$ else $q_{i-t-1} := \lfloor (x_i b + x_{i-1})/y_t \rfloor$;
   3.2 while $(q_{i-t-1}(y_t b + y_{t-1}) > x_i b^2 + x_{i-1} b + x_{i-2})$ do $q_{i-t-1} := q_{i-t-1} - 1$;
   3.3 $x := x - q_{i-t-1} y b^{i-t-1}$;
   3.4 if $(x < 0)$ then $x := x - yb^{i-t-1}$, $q_{i-t-1} := q_{i-t-1} - 1$;
4. $r := x$;
5. return (q, r);

Opomba: Pogoj $n \geq t \geq 1$ lahko zamenjamo s pogojem $n \geq t \geq 0$, če vzamemo $x_j = y_j = 0$, kadar se v algoritmu 11 pojavi indeks $j < 0$.

Če velja $y_t \geq \lfloor b/2 \rfloor$ in b sod, se korak 2 izvede kvečjemu enkrat. Ocena za cifro q_{i-t-1} kvocienta q v koraku 3.1 ni nikoli manjša kot prava vrednost. Če velja $y_t \geq \lfloor b/2 \rfloor$, potem se korak 3.2 ne ponovi več kot dvakrat. Če spremenimo prireditev v koraku 3.1 v

$$q_{i-t-1} := \lfloor (x_i b^2 + x_{i-1} b + x_{i-2})/(y_t b + y_{t-1}) \rfloor,$$

potem je ocena za q_{i-t-1} skoraj vedno pravilna in korak 3.2 se ne ponovi več kot enkrat.

Oceno $y_t \geq \lfloor b/2 \rfloor$ lahko vedno zagotovimo, če zamenjamo par x, y s parom λx, λy, za primerno izbran λ. Kvocient za λx, λy je enak kvocientu za x, y, ostanek pa je pomnožen z λ. Če je osnova b potenca 2, potem naj bo tudi λ potenca 2. Tedaj je množenje z λ enostavno pomik v levo binarne reprezentacije x in y. Postopku množenja z λ pravimo *normalizacija.*

Poglejmo zahtevnost algoritma 11. Korak 3.1 prispeva eno enostavno deljenje, korak 3.2 prispeva dve enostavni množenji, korak 3.3 pa $t + 1$ enostavnih množenj. Če predpostavimo, da normalizacija poveča število cifer v številu x za ena, potem vsaka iteracija koraka 3 prinese $1 + (t + 3) = t + 4$ enostavnih množenj. Zato algoritem 11 z normalizacijo potrebuje $(n - t)(t + 4)$ enostavnih množenj in največ $(n - t)$ enostavnih deljenj.

5.3 Evklidov algoritem

Največji skupni delitelj števil a in b lahko izračunamo tako, da poiščemo razcep števil a in b na potence praštevil. Vendar ta postopek ni učinkovit, saj je problem faktorizacije na praštevila težak problem. Evklidov algoritem (algoritem 12) pa je učinkovit algoritem za izračun največjega skupnega delitelja dveh števil, ki ne potrebuje faktorizacije števil.

ALGORITEM 12 - Evklidov algoritem za izračun največjega skupnega delitelja

Podatki: dve nenegativni celi števili a in b , $a \geq b$.
Rezultat: največji skupni delitelj a in b.

1. while $(b \neq 0)$ do
   $r := a \bmod b$, $a := b$, $b := r$;
2. return (a);

Analizirajmo algoritem 12. Naivno bi lahko rekli, da je časovna zahtevnost Evklidovega algoritma $O(a)$, ki ni polinomska zahtevnost pri podatkih a in b. Vendar bomo pokazali, da je časovna zahtevnost Evklidovega algoritma linearna v lg a in lg b. Označimo število celoštevilskih deljenj Evklidovega algoritma pri podatkih a in b z $E(a, b)$. Naj bo F_n n-to Fibonaccijevo število, definirano s predpisom $F_0 = 0$, $F_1 = 1$ in $F_n = F_{n-1} + F_{n-2}$.

LEMA 5.1 Naj bosta $a > b > 0$ celi števili in naj Evklidov algoritem pri podatkih (a, b) izvede k korakov, torej $E(a, b) = k$. Potem velja $a \geq F_{k+2}$ in $b \geq F_{k+1}$.

DOKAZ Dokaz poteka z indukcijo po k. Naj bo $a_0 = a$ in $a_1 = b$. Trditev velja za $k = 1$. Tedaj $a_0 = q_0 a_1$ in $a_0 > a_1$. Najmanjši a_0 in a_1 dobimo tako, da vzamemo $q_0 = 2$, $a_1 = 1$, sledi $a_0 = 2$. Predpostavimo, da lema velja za vse $i < k$. Potem je prvi korak enak $a_0 = q_0 a_1 + a_2$ in $E(a_1, a_2) = k - 1$. Velja $a_1 \geq F_{k+1}$ in $a_2 \geq F_k$ po indukcijski predpostavki. Sledi $a_0 \geq a_1 + a_2 \geq F_{k+2}$. □

DEFINICIJA Naj bosta a in b celi števili. Pravimo, da je par (a, b) *leksikografsko manjši* od para (a', b'), če je $a < a'$, ali če je $a = a'$ in $b < b'$.

POSLEDICA 5.2 Naj Evklidov algoritem pri podatkih (a, b), $a > b > 0$, izvede k korakov in naj bo par (a, b) leksikografsko najmanjši tak par. Potem $(a, b) = (F_{k+2}, F_{k+1})$.

DOKAZ Po lemi 5.1 velja $a \geq F_{k+2}$ in $b \geq F_{k+1}$. Trdimo, če je $a = F_{k+2}$ in $b = F_{k+1}$, potem Evklidov algoritem izvede k deljenj. Res, Evklidov algoritem nam pri podatkih (F_{k+2}, F_{k+1}) da zaporedje kvocientov $q_0 = 1, \ldots, q_{k-2} = 1, q_{k-1} = 2$, saj za $k \geq 2$ velja F_{k+2} div $F_{k+1} = 1$ in $F_{k-1} = F_{k+1}$ mod F_k . □

POSLEDICA 5.3 Naj bo $a > b > 0$. Potem je $E(a, b) < c_1 \log a + c_2 - 2 + c_3 a^{-1}$, kjer so c_1, c_2 in c_3 konstante neodvisne od a in b. Podobno je $E(a, b) < c_1 \log b + c_2 - 1 + c_3 b^{-1}$.

DOKAZ Predpostavimo, da je $a \geq F_{k+2}$. Ker lahko k-to Fibonaccijevo število eksplicitno zapišemo kot $F_k = (\alpha^k - \beta^k)/\sqrt{5}$, kjer je $\alpha = (1+\sqrt{5})/2$ in $\beta = (1-\sqrt{5})/2$, je

$$a \geq \frac{\alpha^{k+2} - \beta^{k+2}}{\sqrt{5}} \geq \frac{\alpha^{k+2} - 1}{\sqrt{5}}.$$

Zato velja $(k + 2)\log \alpha \leq \log(1 + a\sqrt{5})$. Uporabimo naslednji dve lastnosti logoritmov,

$$\log(x + 1) = \log x + \log(1 + 1/x) \text{ in } \log(1 + 1/x) < 1/x,$$

in dobimo

$$(k + 2)\log \alpha < \log(a\sqrt{5}) + 1/(a\sqrt{5}).$$

Sledi, da je $k < c_1 \log a + c_2 - 2 + c_3 a^{-1}$, kjer $c_1 = 1/\log\alpha$, $c_2 = \log\sqrt{5}/\log\alpha$ in $c_3 = 1/\sqrt{5}\log\alpha$. Obratno, če je $k \geq c_1 \log a + c_2 - 2 + c_3 a^{-1}$, je $a < F_{k+2}$. Vzamemo $k = \lceil c_1 \log a + c_2 - 2 + c_3 a^{-1} \rceil$. Pri tej vrednosti k je $a < F_{k+2}$, zato je po lemi 5.1 vrednost $E(a, b) < k$. Ker je $E(a, b)$ celo število, je $E(a, b) < c_1 \log a + c_2 - 2 + c_3 a^{-1}$. Podobno dokažemo, da je $E(a, b) < c_1 \log b + c_2 - 1 + c_3 b^{-1}$. □

Zadnja posledica nam da polinomsko zahtevnost Evklidovega algoritma pri podatkih (a, b). Vse operacije potekajo na številih $\leq a$. Algoritem opravi $O(\lg a)$ deljenj na številih velikosti $O(\lg a)$, zato je ocena časovne zahtevnosti enaka $O((\lg a)^3)$. Z malo več truda dobimo še boljšo oceno.

TRDITEV 5.4 Evklidov algoritem ima pri podatkih a in b časovno zahtevnost $O((\lg a)(\lg b))$.

DOKAZ Pri deljenju a_i z a_{i+1} dobimo kvocient q_i in ostanek a_{i+2} v času $O((\lg q_i)(\lg a_{i+1}))$. Potem je

$$\sum_{0\le i\le k-1}(\lg q_i)(\lg a_{i+1}) \le (\lg b)\sum_{0\le i\le k-1}(\lg q_i) \le (\lg b)(k + \log_2(\prod_{0\le i\le k} q_i)).$$

Ker je $k = O(\lg a)$ po posledici 5.3 in velja $q_0q_1\cdots q_{k-2}q_{k-1} \le a$, saj je $a = a_0 \ge q_0a_1 \ge q_0q_1a_2 \ge \ldots$, dobimo časovno zahtevnost $O((\lg a)(\lg b))$. □

Evklidov algoritem je moč razširiti tako, da ne vrne le največji skupni delitelj d števil a in b, ampak tudi števili x in y, ki zadoščata Diofantski enačbi $ax + by = d$ (ali kongruenčni enačbi $ax \equiv d \pmod b$). Imenujemo ga kar razširjen Evklidov algoritem (algoritem 13).

ALGORITEM 13 - Razširjen Evklidov algoritem

Podatki: dve nenegativni celi števili a in b , $a \ge b$.
Rezultat: $d = D(a, b)$ in števili x in y, ki zadoščata enačbi $ax + by = d$.

1. if $(b = 0)$ then $d := a,\ x := 1,\ y := 0$;
2. $x_0 := 1,\ y_0 := 0,\ x_1 := 0,\ y_1 := 1$;
3. while $(b \ne 0)$ do
   3.1 $q := \lfloor a/b \rfloor,\ r := a - qb,\ x := x_0 - qx_1,\ y := y_0 - qy_1$;
   3.2 $a := b,\ b := r,\ x_0 := x_1,\ y_0 := y_1,\ x_1 := x,\ y_1 := y$;
4. $d := a,\ x := x_0,\ y := y_0$
5. return (d, x, y);

TRDITEV 5.5 Razširjen Evklidov algoritem pri podatkih a in b, izračuna $d = D(a, b)$ in $x, y \in \mathbf{Z}$, rešitvi enačbe $ax + by = d$, v času $O((\lg a)(\lg b))$.

DOKAZ Zahtevnost običajnega Evklidovega algoritma smo na vsakem koraku povečali za dve množenji in dve odštevanji. Zanima nas kakšna so števila x_i in y_i. Definirajmo $\alpha_i = y_{i+1}x_i - x_{i+1}y_i$. Hitro se lahko prepričamo, da je $\alpha_{i+1} = -\alpha_i$. Ker je $\alpha_0 = 1$, je $\alpha_i = \pm 1$ za vsak i, $1 \le i \le k+1$. Torej velja $D(x_i, y_i) = 1$ za vsak i, $1 \le i \le k + 1$. Ker je $0 = a_{k+1} = ax_{k+1} + by_{k+1}$, velja

$$(a/d)x_{k+1} = -(b/d)y_{k+1},$$

kjer $d = D(a, b)$. Ker je $D(x_{k+1}, y_{k+1}) = 1$ in $D(a/d, b/d) = 1$, velja $x_{k+1} = \pm(b/d)$ in $y_{k+1} = \pm(a/d)$. Ker sta zaporedji $\{|x_i|\}$ in $\{|y_i|\}$ naraščujoči, velja $|x_i| < b$, $|y_i| < a$ za vsak i, $1 \le i \le k + 1$. To pomeni, da se ocena zahtevnosti razširjenega Evklidovega algoritma glede na običajen Evklidov algoritem ne spremeni in je enaka $O((\lg a)(\lg b))$. □

Razširjen Evklidov algoritem se poleg izračuna največjega skupnega delitelja d števil $a, b \in \mathbf{Z}$ in iskanja celoštevilskih rešitev x in y Diofantske enačbe $ax + by = d$, uporablja tudi za izračun multiplikativnih inverzov v $\mathbf{Z}_n^*$, glej §5.4 (iii).

5.4 Osnovne operacije v Z_n

Naj bo n pozitivno celo število. Kot prej, elemente $\mathbf{Z}_n$ predstavljajo števila $\{1, 2, \ldots, n-1\}$. Operaciji, ki izračuna ostanek po modulu n pravimo *modularna redukcija* po modulu n. Klasična metoda za modularno redukcijo je izračun ostanka po modulu n s celoštevilskim deljenjem, vendar obstajajo še druge, npr. Montgomeryjeva redukcija [19, str. 600] in Barrettova redukcija, ki jo bomo opisali v razdelku §5.6. Seštevanje, odštevanje in množenje v $\mathbf{Z}_n$ izvajamo po modulu n. Imenujemo jih *modularno seštevanje*, *modularno odštevanje* in *modularno množenje*. Vidimo, če sta $a, b \in \mathbf{Z}_n$, potem je

$$(a + b) \bmod n = \begin{cases} a+b, & a+b \le n \\ a+b-n, & a+b \ge n \end{cases}.$$

Zato lahko modularno seštevanje (in odštevanje) izvedemo brez zahtevnega deljenja. Modularno množenje a in b izvedemo tako, da enostavno zmnožimo a in b kot cela števila, potem pa rezultat reduciramo po modulu n. Inverze v $\mathbf{Z}_n$ lahko izračunamo z razširjenim Evklidovim algoritmom (glej §5.3). V nadaljevanju bomo poleg naštetih operacij opisali še *modularno eksponiranje*. V tabeli 4 je predstavljena računska zahtevnost osnovnih operacij v $\mathbf{Z}_n$.

Operacija		Zahtevnost
Modularno seštevanje	$(a + b) \bmod n$	$O(\lg n)$
Modularno odštevanje	$(a - b) \bmod n$	$O(\lg n)$
Modularno množenje	$(a \cdot b) \bmod n$	$O((\lg n)^2)$
Modularno invertiranje	$a^{-1} \bmod n$	$O((\lg n)^2)$
Modularno eskponiranje	$a^k \bmod n, k < n$	$O((\lg n)^3)$

TABELA 4: Računska zahtevnost osnovnih operacij v $\mathbf{Z}_n$.

(i) Modularno seštevanje in odštevanje

Kot pri celih številih, sta tudi pri številih ostankov po modulu n operaciji seštevanja in odštevanja najenostavnejši. Če sta x in y nenegativni števili, kjer $x, y < n$, potem velja naslednje:
(i) $x + y < 2n$
(ii) če je $x \ge y$, potem velja $0 \le x - y < n$
(iii) če je $x < y$, potem velja $0 \le x + n - y < n$

Torej, če sta $x, y \in \mathbf{Z}_n$, potem izvedemo natančno seštevanje (algoritem 7) z dodatnim korakom, če je le-ta potreben, tj. odštejemo n, če je $x + y \ge n$. Modularno odštevanje opravimo z algoritmom 8 pri predpostavki $x \ge y$.

(ii) Modularno množenje

Modularno množenje je zahtevnejše od množenja, saj zahteva tako natančno množenje (algoritem 9) kot tudi metodo za modularno redukcijo. Najbolj očitna metoda modularne redukcije je izračun ostanka pri deljenju s številom n z uporabo natančnega deljenja (algoritem 11). To je tako imenovani *klasični algoritem* za izvedbo modularnega množenja.

ALGORITEM 14 - Klasično modularno množenje

Podatki: dve pozitivni celi števili x, y in modul n.
Rezultat: $r = x \cdot y \bmod n$.

1. izračunaj $x \cdot y$ (algoritem 9)
2. izračunaj ostanek r, ko $x \cdot y$ deliš z n (algoritem 11).
3. return (r);

Recimo, da sta $x, y \in \mathbf{Z}_n$, $0 \le x, y \le n - 1$, $k = \lfloor \log_2 n \rfloor + 1$. Potem lahko izračunamo $x \cdot y \bmod n$ tako, da najprej izračunamo $2k$ bitni produkt $x \cdot y$, potem pa ga reduciramo po modulu n. Ta dva koraka lahko naredimo v času $O(k^2) = O((\lg n)^2)$.

(iii) Modularno invertiranje

Za implementacijo RSA potrebujemo tudi operacijo modularnega inventiranja in sicer pri izračunu dešifrirnega eksponenta. Po trditvi 2.9 je enačba $ax \equiv 1 \pmod n$ rešljiva, če je $D(a, n) = 1$. Rešitev je enolično določena po modulu n. Torej, če je $D(a, n) = 1$, lahko poiščemo multiplikativen inverz za a po modulu n. Izračun inverzov v $\mathbf{Z}_n$ opravi algoritem 15.

ALGORITEM 15 - Izračun multiplikativnih inverzov v $\mathbf{Z}_n$

Podatki: $a \in \mathbf{Z}_n$.
Rezultat: $a^{-1} \bmod n$, če le-te obstaja.

1. uporabi razširjen Evklidov algoritem (algoritem 13) in poišči števili x in y, rešitvi enačbe $ax + ny = d$, kjer je $d = D(a, n)$.
2. if ($d > 1$) then "$a^{-1} \bmod n$ ne obstaja" else return (x);

Časovno zahtevnost algoritema 15 je $O((\lg n)^2)$, saj uporabimo le razširjen Evklidov algoritem.

(iv) Modularno eksponiranje

Poglejmo si še modularno eksponiranje, tj. izračun $a^k \bmod n$. Označimo dolžino n v binarni reprezentaciji z l ($l = \lfloor \log_2 n \rfloor + 1$). Kot smo že povedali sta operaciji šifriranja in dešifriranja v RSA kriptosistemu modularni eksponiranji. Izračun $a^k \bmod n$ lahko naredimo kot $k - 1$ modularnih množenj, vendar to je zelo neučinkovito, če je eksponent k veliko število (k je lahko tudi $\phi(n) - 1$, ki pa je eksponentno velik glede na l).

Modularno eksponiranje lahko učinkovito izvedemo z uporabo metode kvadriraj-in-množi, ki je pomembena za več kriptografskih sistemov. Ena verzija te metode je algoritem 16. Temelji na naslednji enakosti. Naj bo binarna reprezentacija za k enaka $\sum_{i=0}^{t-1} k_i 2^i$, kjer $k_i \in \{0, 1\}$. Potem je

$$a^k = \prod_{i=0}^{t} a^{k_i 2^i} = (a^{2^0})^{k_0} (a^{2^1})^{k_1} \ldots (a^{2^t})^{k_t} .$$

ALGORITEM 16 - Algoritem kvadriraj-in-množi za eksponiranje v $\mathbf{Z}_n$

Podatki: $a \in \mathbf{Z}_n$ in število k, ki ima binarno reprezentacijo $k = \sum_{i=0}^{t-1} k_i 2^i$.
Rezultat: $b := a^k \bmod n$.

1. $b := 1$, if $(k = 0)$ then return (b);
2. $A := a$;
3. if $(k_0 = 1)$ then $b := a$;
4. for $i := 1$ to $(t-1)$ do
   4.1 $A := A^2 \bmod n$;
   4.2 if $(k_i = 1)$ then $b := A \cdot b \bmod n$;
5. return (b);

Pri metodi kvadriraj-in-množi se število modularnih množenj, potrebnih za izračun $a^k \bmod n$, zmanjša na največ $2t$, kjer je t število bitov binarne reprezentacije eksponenta k. Bolj natančno, imamo $t - 1$ kvadriranj (algoritem 10) in $w(k) - 1$ množenj (algoritem 9), kjer je $w(k) \le t$ število enic v binarni reprezentaciji k. V povprečju velja $w(k) = t/2$. Če je kvadriranje približno enako zahtevno kot množenje, potem je pričakovana zahtevnost eksponiranja približno $3/2 \cdot \lg k$ množenj. Ker imamo $O(\lg k)$ kvadriranj in množenj števil velikosti $O(\lg n)$ in je $k \le n$, lahko $a^k \bmod n$ izračunamo v času $O((\lg k)(\lg n)^2) = O((\lg n)^3)$. Rezultat nam pove, da je časovna zahtevnost RSA šifriranja/dešifrirnja polinomska v $\lg n$, kjer je n RSA modul.

5.5 Uporaba Kitajskega izreka o ostankih

Implementacije RSA kriptosistema v primeru dešifriranja in generiranje podpisov pogosto uporabijo Kitajski izrek o ostankih (izrek 2.11) za hitrejši izračun sporočila $m = c^d \bmod n$ (glej §3.3) oziroma podpisa $s = \tilde{m}^d \bmod n$ (glej §3.4) Namesto računanja po modulu n, računamo po modulu p in q.

ALGORITEM 17 - Uporaba kitajskega izreka o ostankih

Podatki: zasebni RSA ključ (p, q, d) in šifrirano sporočilo c.
Rezultat: sporočilo $m \equiv c^d \pmod n$.

1. $c_p := c \bmod p$; $c_q := c \bmod q$;
2. $d_p := d \bmod (p-1)$; $d_q := d \bmod (q-1)$;
3. $m_p := (c_p)^{d_p} \bmod n$; $m_q := (c_q)^{d_q} \bmod n$;
4. $T_1 := q^{p-1} \bmod n$; $T_2 := p^{q-1} \bmod n$;
5. $m_1 := T_1 m_p \bmod n$; $m_2 := T_2 m_q \bmod n$;
6. $m := m_1 + m_2$;
7. if $(m > n)$ then $m := m - n$;
6. return (m);

Delovni čas izračuna $m = T_1 m_p + T_2 m_q$ (korak 5 in 6), kjer sta T_1 in T_2 koeficienta rešitve iz Kitajskega izreka o ostankih, je zanemarljivo majhen v primerjavi z modularnima eksponiranjima. Ker sta praštevili p in q polovico krajši od modula n in ker množenje zahteva kvadratičen čas, je množenje po modulu p štirikrat hitrejše kot po modulu n. Prav tako je d_p za polovico krajši od d, zato je izračun $(c_p)^{d_p} \bmod p$ osemkrat hitrejši od izračuna $c^d \bmod n$. Dešifriranje je v celoti hitrejše za štirikrat. Ocena časovne zahtevnosti algoritma 17 ostane enaka $O((\lg n)^3)$.

5.6 Barrettova modularna redukcija

Učinkovitejša metoda za izračun modularne redukcije od izračuna ostanka pri celoštevilskem deljenju (algoritem 11) je Barrettova modularna redukcija (algoritem 18). Barrettova redukcija izračuna $r = x \bmod n$ pri danem številu x in modulu n. Algoritem zahteva izračun vrednosti $\mu = \lfloor b^{2k}/n \rfloor$, kjer je k dolžina modula n pri osnovi b. Algoritem je koristen v primeru, ko imamo veliko modularnih redukcij z enim samim modulom, npr. pri RSA (de)šifriranju. Izračun μ zahteva fiksno mnogo dela in je zanemarljiv v primerjavi z modularnim eksponiranjem. Velikost osnove b je ponavadi blizu velikosti besede procesorja, zato predpostavimo $b > 3$.

ALGORITEM 18 - Barrettova modularna redukcija

Podatki: pozitivni števili $x = (x_{2k-1}\ldots x_1x_0)_b$ in $n = (n_{k-1}\ldots n_1n_0)_b$, kjer $n_{k-1} \neq 0$ in $\mu = \lfloor b^{2k}/n \rfloor$.
Rezultat: $r := x \bmod n$.

1. $q_1 := \lfloor x/b^{k-1} \rfloor$, $q_2 := q_1 \cdot \mu$, $q_3 := \lfloor q_2/b^{k+1} \rfloor$;
2. $r_1 := x \bmod b^{k+1}$, $r_2 := q_3 \cdot n \bmod b^{k+1}$, $r := r_1 - r_2$;
3. if $(r < 0)$ then $r := r + b^{k+1}$;
4. while $(r > 0)$ do $r := r - n$;
5. return (r);

Kot vemo, nam celoštevilsko deljenje števila x s številom n natanko določi števili Q in R, da velja $x = Qn + R$, kjer $0 \leq R < n$. Algoritem 18 temelji na tem, da lahko kvocient $Q = \lfloor x/n \rfloor$ zapišemo kot $Q = \lfloor (x/b^{k-1})(b^{2k}/n)(1/b^{k+1}) \rfloor$ in ga aproksimiramo s številom $q_3 = \lfloor \lfloor x/b^{k-1} \rfloor \cdot \mu/b^{k+1} \rfloor$. Označimo $x_1 = x/b^{k-1}$ in $x_2 = b^{2k}/n$. Potem velja ocena $(x_1 - 1)(x_2 - 1) < q_1\mu \leq x_1x_2$. Zato je

$$0 \leq x_1x_2 - q_2 < x_1 + x_2 - 1 < 2b^{k+1}.$$

kjer upoštevamo $x < b^{2k}$ in $n > b^{k-1}$. Ker je $Q = x_1x_2 \operatorname{div} b^{k+1}$ in $q_3 = q_2 \operatorname{div} b^{k+1}$, je

$$(Q - q_3)b^{k+1} = x_1x_2 - q_2 - (x_1x_2 - q_2) \bmod b^{k+1} < 2b^{k+1},$$

oziroma $Q - q_3 < 2$. Pokazali smo, da število q_3 v koraku 1 zadošča neenakosti $Q - 2 < q_3 \leq Q$. V koraku 2 je $0 \leq r_1, r_2 < b^{k+1}$, zato je $-b^{k+1} < r_1 - r_2 < b^{k+1}$. Velja $r_1 - r_2 \equiv (Q - q_3)n + R \pmod{b^{k+1}}$. Če upoštevamo oceni $n < b^k$ in $3 < b$, velja

$$0 \leq (Q - q_3)n + R < 3n < b^{k+1}.$$

Zato, če je razlika $r_1 - r_2 \geq 0$, potem je $r_1 - r_2 = (Q - q_3)n + R$, sicer pa je razlika $r_1 - r_2 < 0$ in velja $r_1 - r_2 + b^{k+1} = (Q - q_3)n + R$. V obeh primerih, se korak 4 ponovi največ dvakrat, saj je $0 \leq r < 3n$.

Vsa deljenja v algoritmu 18 so enostavno pomiki v desno pri osnovi b. Vrednost q_2 potrebujemo le za izračun vrednosti q_3. Ker $k + 1$ cifer z najnižjim redom števila q_3 ne potrebujemo, je dovolj izvesti le delno natančno množenje pri izračunu produkta $q_1 \cdot \mu$. Edini vpliv teh cifer je prenos iz mesta $k + 1$ na mesto $k + 2$. Če je osnova b dovolj velika v primerjavi s številom k, lahko ta prenos izračunamo tako, da pri računanju upoštevamo le mesti k in $k + 1$. Zato $k - 1$ cifer števila q_2 z najnižjim redom ni potrebno izračunati. Ker sta dolžini števil q_1 in μ največ $k + 1$, potrebujemo za izračun števila q_3 največ

$$(k + 1)^2 - \frac{k(k-1)}{2} = (k^2 + 5k + 2)/2$$

enostavnih množenj.

5.7 Miller-Rabinov test

V praksi želimo za faktorja RSA modula veliki (npr. 512-bitni) praštevili p in q. Izbrati moramo naključno liho celo število izbrane dolžine in preveriti, ali je praštevilo. Število praštevil dolžine 512 bitov je $\pi(2^{512}) - \pi(2^{511})$. Če upoštevamo izrek o gostoti praštevil (izrek 2.6) je verjetnost, da je 512-bitno število praštevilo približno

$$2 \cdot \frac{\left(\frac{2^{512}}{\ln 2^{512}} - \frac{2^{511}}{\ln 2^{511}}\right)}{2^{512} - 2^{511}} \approx 0{,}00562\ldots,$$

kjer množimo z 2 zato, ker gledamo le liha števila. Kako pa preverimo, ali je izbrano število praštevilo? Miller-Rabinov test je verjetnostni algoritem za preverjanje praštevilskosti, znan tudi kot *test krepkih psevdopraštevil*. Povod naslednji definiciji je trditev 2.25.

DEFINICIJA Naj bo n liho praštevilo in velja $n - 1 = 2^s r$, kjer je r liho število. Naj bo a celo število na intervalu $[1, n-1]$. Potem definiramo naslednje.

(i) Če je $a^r \not\equiv 1 \pmod{n}$ in $a^{2^j r} \not\equiv -1 \pmod{n}$ za vsak j, $0 \le j \le s-1$, potem število a imenujemo *krepka priča* (sestavljenosti) za n.

(ii) Sicer, če je $a^r \equiv 1 \pmod{n}$ ali $a^{2^j r} \equiv -1 \pmod{n}$ za nek j, $0 \le j \le s-1$, potem a imenujemo *krepek lažnivec* (praštevilskosti) za n, n pa imenujemo *krepko psevdopraštevilo k osnovi a*.

Za liho sestavljeno število n velja, da je največ 1/4 vseh števil a, $1 \le a \le n-1$, krepkih lažnivcev [17, str. 117]. Velja še boljša ocena. Če je $n \ne 9$, potem je število krepkih lažnivcev za število n največ $\phi(n)/4$, kjer je ϕ Eulerjeva funkcija.

ALGORITEM 19 - Miller-Rabinov test

Podatki: liho celo število $n \ge 3$ in parameter $t \ge 1$.
Rezultat: odgovor "praštevilo" ali "sestavljeno število" na vprašanje "Ali je število praštevilo?".

1. zapiši $n - 1 = 2^s r$, kjer je r liho število.
2. for $i := 1$ to t do
   2.1 izberi naključno število a, $2 \le a \le n-2$.
   2.2 $y := a^r \bmod n$;
   2.3 if $(y \ne 1)$ and $(y \ne n-1)$ then
   $j := 1$;
   while $(j \ge s-1)$ and $(y \ne n-1)$ do
   $y := y^2 \bmod n$;
   if $(y = 1)$ then return ("praštevilo");
   $j := j + 1$;
   if $(y \ne n-1)$ then return ("sestavljeno število");
3. return ("praštevilo");

Miller-Rabinov test (algoritem 19) preveri, ali izbrano število a zadošča pogojem krepke priče. Če je v *while* zanki $y = 1$ (točka 2.3), potem je

$$a^{2^j r} \equiv 1 \pmod{n} \quad \text{in} \quad a^{2^{j-1} r} \not\equiv \pm 1 \pmod{n}.$$

Po trditvi 2.24 je n sestavljeno število ($D(a^{2^{j-1} r} - 1, n)$ je netrivialen faktor za n). Ko algoritem izstopi iz zanke, nam pogoj $y \ne n-1$ pravi, da je a krepka priča za n.

Vidimo torej, če Miller-Rabinov algoritem odgovori s "sestavljeno število", potem je število n gotovo sestavljeno število, ker praštevila ne kršijo trditve 2.25. Podobno, če je število n res praštevilo, potem algoritem vedno odgovori s "praštevilo". Po drugi strani, če je število n v resnici sestavljeno število, potem je verjetnost napačnega odgovora "praštevilo" Miller-Rabinovega algoritma pri parametru t manjša od $(1/4)^t$. V resnici je verjetnost napake pri večini števil veliko manjša od $(1/4)^t$, saj je število krepkih lažnivcev ponavadi veliko manjše od $\phi(n)/4$. Časovna zahtevnost algoritma 19 je $O((\lg n)^3)$, saj uporabimo eno modularno eksponiranje in največ $O(\lg n)$ modularnih kvadriranj v času $O((\lg n)^2)$.

Poleg Miller-Rabinovega testa omenimo še dva znana praštevilska testa, Fermatov test [19, str. 136] in Solovay-Strassen test [19, str. 137]. Slednji je, tako kakor Miller-Rabinov test, verjetnostni test tipa Monte-Carlo (glej §2.3). Oba algoritma odgovorita pravilno v primeru, da je število res praštevilo ali, če odgovorita "sestavljeno število". Vendar pa ima Miller-Rabinov test manjšo napako in je manj zahteven pri implementaciji.

DEFINICIJA Številu n, za katerega verjamemo, da je praštevilo na podlagi verjetnostnega praštevilskega testa, pravimo *verjetno praštevilo*.

Poznamo tudi t.i. prave praštevilske teste, kjer je število, ki ga označijo za praštevilo, gotovo praštevilo, in tudi algoritme za generiranje tako verjetnih kot tudi pravih naključnih praštevil. Ogledali si bomo preprost algoritem (algoritem 20), za genereranje naključnih praštevil, ki temelji na Miller-Rabinovem testu praštevilskosti.

ALGORITEM 20 - Naključno iskanje praštevil z Miller-Rabinovim testom

Podatki: izbrana bitna dolžina k praštevila in parameter $t \geq 1$.
Rezultat: naključno verjetno praštevilo dolžine k bitov.

1. generiraj naključno liho število n dolžine k bitov.
2. uporabi poskusno deljenje z lihimi praštevili $\leq B$. Če je n deljiv, pojdi na korak 1.
3. če Miller-Rabinov test pri n in t odgovori "praštevilo" return (n), sicer pojdi na korak 1.

V algoritmu delimo naključno liho število z vsemi praštevili manjšimi od nekega števila B. Ta praštevila imamo vnaprej spravljena v tabeli. Zgornjo mejo B izberemo tako, da je $B = E/D$, kjer je E čas, ki ga potrebujemo za modularno eksponiranje največjega k-bitnega števila in D čas, ki ga potrebujemo, da preverimo, ali je majhno praštevilo delitelj tega števila. Ker je B odvisen od implementacije operacij za dolga števila, se ponavadi določi eksperimentalno. Za $B = (\lg n)^2/\lg\lg n$ je pričakovano število klicev Miler-Rabinovega testa omejeno z $O(\lg n)$, glej [1, str. 295], zato je ocena časovne zahtevnosti algoritma 20 enaka $O((\lg n)^4)$.

V praksi želimo, da verjetnost napake pri generiranju verjetnih praštevil ni večja od $(1/2)^{80}$. V tabeli 5 so navedene vrednosti za parameter t pri dani bitni dolžini k modula n, za katere je verjetnost napake $\leq (1/2)^{80}$ [19, str. 148].

k	t	k	t	k	t	k	t	k	t
100	27	300	9	500	6	700	4	900	3
150	18	350	8	550	5	750	4	950	3
200	15	400	7	600	5	800	4	1000	3
250	12	450	6	650	4	850	3	1050	3

TABELA 5: Pri izbranem k najmanjši t, pri katerem je verjetnost napake $\leq (1/2)^{80}$

5.8 SHA-1

SHA-1 (Secure Hash algorithm) je zgoščevalna funkcija, ki jo je predlagala NIST (U.S. National Institute for Standards and Technology) za aplikacije v nekaterih državnih ustanovah [19, str. 348]. Je ena tistih zgoščevalnih funkcij (brez ključa), ki so jo izdelali "iz nič" za točno določen namen stiskanja z optimiziranim učinkom. Uporabna je za programsko implementacijo na 32 bitnih računalnikih. V našem primeru jo bomo uporabili pri shemi elektronskega podpisa IFSSA, ki jo bomo opisali v naslednjem razdelku (§5.9).

Glavne značilnosti zgoščevalne funkcije SHA-1 so:

- Zgoščena vrednost je dolžine 160 bitov.
- Uporabljenih je pet 32-bitnih verižnih spremenljivk.
- Zgoščevalna funkcija ima štiri iteracije. Funkcije *f*, *g* in *h* (glej tabelo 6) so uporabljene v naslednjem vrstnem redu: najprej *f*, potem *h*, sledi *g* in spet *h*. Vsaka iteracija ima 20 korakov.
- Znotraj zgoščevalne funkcije je vsak blok 16 besed razširjen na blok 80 besed, tako da je vsaka od zadnjih 64 besed XOR štirih prejšnjih besed v razširjenem bloku. Teh 80 besed je potem ena za drugo vstavljena v 80 korakov.
- Glavni korak poleg funkcij *f*, *g* in *h* vsebuje še konstantno 5-bitno in 30-bitno rotacijo.
- SHA-1 uporablja štiri neničelne aditivne konstante.

Notacija	Pomen
u, v, w	32-bitne spremenljivke
0x67452301	šestnajstiško 32-bitno število (byte z najnižjim redom: 01)
$+$	seštevanje po modulu 2^{32}
$\bar{u}$	bitni komplement
$u \hookleftarrow s$	u rotiran v levo za s bitov
uv	bitna operacija IN (AND)
$u \vee v$	bitna operacija inkluzivni-ALI (OR)
$u \oplus v$	bitna operacija ekskluzivni-ALI (XOR)
$f(u, v, w)$	$uv \vee \bar{u}v$
$g(u, v, w)$	$uv \vee uw \vee vw$
$h(u, v, w)$	$u \oplus v \oplus w$
$(X_1, \ldots, X_j) := (Y_1, \ldots, Y_j)$	večkratna prireditev $(X_i := Y_i)$

TABELA 6: Notacija za SHA-1 algoritem

Niz	Zgoščena vrednost
""	DA39A3EE 5E6B4B0D 3255BFEF 95601890 ADF80709
"a"	86F7E437 FAA5A7FC E15D1DDC B9EAEAEA 377667B8
"abc"	A9993E36 4706816A BA3E2571 7850C26C CD0D89D
"abcdefghijklmnopqrstuvwxyz"	F71C2710 9C692C1B 56BBDCEB 5B9D2865 B3708DBC

TABELA 7: Testni vektorji za SHA-1.

ALGORITEM 21 – SHA-1 algoritem

Podatki: niz bitov x dolžine $l \geq 0$ (za notacijo glej tabelo 6)
Rezultat: 160 bitna zgoščena vrednost za x (testni vektorji so v tabeli 7)

1. *Definicija konstant.* Definiraj pet 32-bitnih začetnih verižnih vrednosti (začetnih vektorjev):
   h_1 := 0x67452301;
   h_2 := 0xEFCDAB89;
   h_3 := 0x98BADCFE;
   h_4 := 0x10325476;
   h_5 := 0xC3D2E1F0;
   Definiraj aditivne konstante za vsako rundo:
   y_1 := 0x5A827999;
   y_2 := 0x6ED9EBA1;
   y_3 := 0x8F1BBCDC;
   y_4 := 0xCA62C1D6;
2. *Predpriprava.* Razširi niz x, da bo njegova dolžina večkratnik števila 512, in sicer takole. Dodaj en sam bit 1. Nadalje dodaj $r - 1$ (≥ 0) bitov 0 za najmanjši r, da bo dolžina niza 64 manj kot večkratnik 512. Dodaj 64-bitno reprezentacijo števila l mod 2^{64} kot dve 32-bitni besedi, najprej tisto z najvišjim redom, potem pa tisto z najnižjim redom. Naj bo m število 512-bitnih blokov v nastalem nizu ($l + r + 64 = 512 \cdot m = 32 \cdot 16 \cdot m$). Podatek $16 \cdot m$ 32-bitnih besed je določen z nizom $x_0 x_1 \ldots x_{16m-1}$. Inicializiraj verižne spremenljivke:
   $(H_1, H_2, H_3, H_4, H_5) := (h_1, h_2, h_3, h_4, h_5)$.
3. *Računski del.* Za vsak i, $0 \leq i \leq m - 1$, kopiraj i-ti blok 32-bitnih besed v začasni spomin:
   for $j := 0$ do 15 do
   $X_j := x_{i+j}$;
   // razširi 16 besed v 80 besed
   for $j := 16$ to 79 do
   $X_j := ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \lhd 1)$.
   // inicializiraj delavne spremenljivke
   $(A, B, C, D, E) := (H_1, H_2, H_3, H_4, H_5)$
   // uporabi vrednosti X_j v naslednjih štirih korakih
   (korak 1) for $j := 0$ to 19 do
   $t := ((A \lhd 5) + f(B, C, D) + E + X_j + y_1)$;
   $(A, B, C, D, E) := (t, A, B \lhd 30, C, D)$;
   (korak 2) for $j := 20$ to 39 do
   $t := ((A \lhd 5) + h(B, C, D) + E + X_j + y_2)$;
   $(A, B, C, D, E) := (t, A, B \lhd 30, C, D)$;
   (korak 3) for $j := 40$ to 59 do
   $t := ((A \lhd 5) + g(B, C, D) + E + X_j + y_3)$;
   $(A, B, C, D, E) := (t, A, B \lhd 30, C, D)$;
   (korak 4) for $j := 60$ to 79 do
   $t := ((A \lhd 5) + h(B, C, D) + E + X_j + y_4)$;
   $(A, B, C, D, E) := (t, A, B \lhd 30, C, D)$;
   // posodobi verižne spremenljivke
   $(H_1, H_2, H_3, H_4, H_5) := (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E)$
4. *Zaključek.* Zgoščena vrednost je $H_1 \| H_2 \| H_3 \| H_4 \| H_5$, kjer $\|$ pomeni lepljenje blokov.

5.9 IFSSA

IFSSA (Integer Factorization Signature Scheme with Appendix) je shema elektronskega podpisa, ki temelji na RSA kriptosistemu in je del standarda IEEE P1363. Nastal je pred kratkim pod okriljem IEEE Standard Department [14]. IFSSA je shema elektronskega podpisa z dodatkom. Torej ima lastnost, da zahteva originalno sporočilo pri procesu preverjanja. Uporablja zgoščevalno funkcijo SHA-1 (§5.8). Opišimo shemo IFSSA.

ALGORITEM 22 – IFSSA

Povzetek: Oseba A podpiše sporočilo $m \in M$ in pošlje podpis s in sporočilo m osebi B. Oseba B preveri pristnost sporočila m s podpisom s.

1. *Generiranje podpisa*. Oseba A naj stori naslednje:
   - Naj bo $l = k - 1$, kjer je k dolžini modula n v bitih. Če je $l < 191$, izbere večji modul n.
   - Izračuna $H = h(m)$, kjer je h zgoščevalna funkcija SHA-1.
   - Naj bo P_1 enak oktetu $4b$, če m prazen niz, in $6b$ sicer.
   - Naj bo P_2 niz $\lfloor (l + 1)/8 \rfloor - 24$ oktetov bb.
   - Zlepi $\tilde{m} = P_1 \| P_2 \| ba \| H \| 33 \| cc$, kjer so ba, 33 in cc okteti.
   - Izračuna $s = \tilde{m}^d \bmod n$. Podpis sporočila m osebe A je s.
2. *Potrditev*. Oseba B dobi sporočilo m in preveri pristnost sporočila m s podpisom s takole:
   - Izračuna $\tilde{m} = s^e \bmod n$.
   - Če je m prazen niz, naj bo P_1 enak oktetu $4b$, sicer $6b$.
   - Preveri, ali je najbolj levi oktet $\tilde{m}$ enak P_1, naslednjih $\lfloor (l + 1)/8 \rfloor - 24$ oktetov enakih bb in oktet za njimi enak ba. Če ni, sporočilo zavrne.
   - Preveri, ali je najbolj desni oktet $\tilde{m}$ enak cc in oktet ob njem enak 33. Če ni, sporočilo zavrne.
   - Odstrani $\lfloor (l + 1)/8 \rfloor - 22$ najbolj levih in 2 najbolj desna okteta iz $\tilde{m}$ in dobi H'.
   - Izračuna $H = h(m)$, kjer je h zgoščevalna funkcija SHA-1.
   - Če je $H = H'$ sporočilo sprejme, sicer ga zavrne.

Opomba: V algoritmu 22 pomeni || operacija lepljenja binarnih nizov, izraz *oktet* pa binarni niz osmih bitov, ki je predstavljen kot šestnajstiško število dolžine 2. Imamo tudi omejitev na bitno dolžino l sporočila m, $l < 2^{61} - 1$, ki je posledica omejitve zgoščevalne funkcije SHA-1.

5.10 RSA Encrypter&Signer v1.0

Program "RSA Encrypter & Signer v1.0" je bil napisan za to diplomsko delo in je priložen na CD-ROM-u. RSA Encrypter & Signer (RSAE.exe) je 32-bitni program za šifriranje in generiranje elektronskih podpisov v okolju Windows 9x. Narejen je s programom Inprise Delphi 5. Program RSAE je poizkus učinkovite implementacije RSA kripto-sistema. Uporabil sem večino (in še nekaj dodatnih) algoritmov predstavljenih v tem poglavju. Glavne funkcije programa RSAE so generiranje RSA ključev s pomočjo Miller-Rabinovega testa (§5.7), šifriranje po RSA šifrirni shemi (§3.3) in elektronski podpis po shemi IFSSA (§5.9), ki je del standarda IEEE P1363 [14] in vsebuje zgoščevalno funkcijo SHA-1 (§5.8). Program je primeren tudi kot nadomestek urejevalnika teksta Notepad. Program se avtomatično namesti (SetupRSAE.exe), vsebuje Pomoč (RSAE.hlp) in se po želji tudi automatično odstrani (Uninstall.exe). Program je brezplačen (Freeware). Na CD-ROM-u je dodana tudi izvorna koda. Več informacij v datotekah Preberi.txt in Licenca.txt.

6. RSA V PRAKSI

V zadnjem poglavju bomo obravnavali probleme, s katerimi se srečujemo pri implementaciji RSA kriptosistema v RSA šifrirno shemo in shemo RSA elektronskega podpisa ter podali ustrezne rešitve. Najvidnejšo vlogo pri uporabi kriptografije v praksi imajo prav gotovo patenti in standardi. Omenili bomo nekaj patentov in standardov, ki so povezani z RSA kriptosistemom. Najprej pa si poglejmo, kakšne so možnosti uporabe RSA kriptosistema v praksi.

RSA kriptosistem je najbolj razširjen kriptosistem z javnimi ključi. Njegove implementacije se pojavljajo tako v programski kot tudi v strojni opremi. RSA kriptosistem se uporablja npr. v spletnih strežnikih in brskalnikih (Netscape, Internet Explorer), v povezavi z elektronsko pošto, elektronskim bančništvom, vgrajen je v pametne kartice [15], npr. za dvig gotovine iz bankomata.

RSA lahko implementiramo tako, da ugotavljamo pristnost pri dostopu uporabnikov na lokalni mreži. Obstajajo različne implementacije že na tako rekoč vseh večjih operacijskih sistemih, kot so Windows (NT, 2000), Unix (Solaris, HP-UX, AIX), Novell Netware in drugi.

Potreba za ugotavljanje pristnosti oddaljenih uporabnikov postaja v času domačih pisarn, poslovnih potovanj in mobilih delavcev, ki vse bolj in bolj potrebujejo oddaljen dostop do informacijskih sredstev, vedno bolj pomembna. RSA lahko uporabimo za nadzor sistemov na telefonski klic ali internetnih povezav in preprečevanje nedovoljenih dostopov.

Če RSA uporabimo v povezavi z zasebnimi šiframi, dobimo t.i. dvostopenjsko ugotavljanje pristnosti. Uporablja se lahko pri zaščiti pomembnih aplikacij, zbirk podatkov, datotek ali spletnih strani, pri delitvi zelo vrednih in/ali nadzorovanih informacij (v notranji ali zunanji mreži) s poslovnimi partnerji in potrošniki in nasploh pri opravljanju elektronskega poslovanja. Uporabimo ga lahko npr. v finančnih spletnih aplikacijah, kot so spletne banke in spletne borze in poskrbimo, da prihaja do poslovne transakcije le z dovoljenimi uporabniki.

6.1 RSA šifririranje v praksi

Obstaja veliko načinov, kako pospešiti RSA šifriranje in dešifriranje v programski in strojni implementaciji. Vendar je tudi s temi izboljšavami RSA šifriranje in dešifriranje še vedno občutno počasnejše (približno 1000-krat) od bolj razširjenih kriptosistemov s simetričnimi ključi, kot je npr. DES [23, str. 70]. V praksi je RSA najpogosteje uporabljen za prenos simetričnih ključev in za šifriranje majhnih podatkov.

(i) Izbira velikosti modula

Če nasprotnik faktorizira javni modul n, potem lahko izračuna $\phi(n)$ in z razširjenim Evklidovim algoritmom (§5.3) dobi d kot rešitev enačbe $ed \equiv 1 \pmod{\phi(n)}$. To pomeni popoln zlom sistema. Izognemo se mu tako, da izberemo taki praštevili p in q, da je faktorizacija RSA modula n neizvedljiva naloga. Glede na razvoj faktorizacijskih algoritmov za faktorizacijo velikih števil je 512 bitni modul n postal premalo varen za napad skupine ljudi, ki so pripravljeni vložiti več sredstev. Od leta 1999 je priporočljivo uporabiti vsaj 768-bitni modul za osebno uporabo, vsaj 1024 bitni modul za poslovno uporabo in vsaj 2048-bitni modul za zelo pomembne podatke oziroma dolgoročno varnost. S tem se izognemo nevarnosti, ki ga predstavljajo algoritmi za faktoriziranje, kot so kvadratično sito, algoritem eliptične krivulje in GNFS (glej §4.1 (i)).

(ii) Izbira praštevil

Kot smo že omenili, moramo praštevili p in q izbrati tako, da je faktorizacija $n = pq$ računsko-časovno neizvedljiva. Največja omejitev pri izbiri p in q, da se izognemo faktorizacijskemu algoritemu eliptične krivulje, je ta, da morata biti praštevili p in q približno iste dolžine (v bitih) in dovolj veliki. Npr. če želimo uporabiti 1024-bitni modul, moramo uporabiti p in q dolžine okrog 512 bitov. Naslednja omejitev na praštevilih je, da razlika $p - q$ ne sme biti premajhna. Če je razlika $p - q$ majhna, potem je $p \approx q$ in zato $p \approx \sqrt{n}$. Tako lahko n faktoriziramo učinkovito tako, da poizkušamo deliti n s števili blizu $\sqrt{n}$. Če sta p in q izbrana naključno, potem bo razlika $p - q$ z veliko verjetnostjo precej velika. Poleg teh omejitev nekateri avtorji predlagajo, da naj bosta p in q tako imenovani *močni praštevili*.

DEFINICIJA Praštevilu p rečemo *močno* praštevilo, če zadošča naslednjim trem pogojem:
1. $p - 1$ ima velik praštevilski faktor, označimo ga z r.
2. $p + 1$ ima velik praštevilski faktor.
3. $r - 1$ ima velik praštevilski faktor.

Razlog za prvi pogoj je, da se izognemo Pollardovemu $p - 1$ algoritmu. S pogojem (2) se izognemo Williamsovemu $p + 1$ faktorizacijskem algoritmu. Pogoj (3) pa poskrbi, da spodleti že prej opisani ciklični napad (§4.4 (iv)).

(iii) Majhen šifrirni eksponent

Če je šifrirni eksponent izbran naključno, potem RSA šifriranje z algoritmom kvadriraj-in-množi (algoritem 16) uporabi k modularnih kvadriranj in približno $k/2$ (manj z dodatno optimizacijo) modularnih množenj, kjer je k dolžina modula n v bitih. Šifriranje pohitrimo tako, da izberemo majhen šifrirni eksponent ali pa takega z malo enicami v binarni reprezentaciji.

V praksi je velikokrat uporabljen šifrirni eksponent $e = 3$. V tem primeru je pomembno, da niti $p - 1$ niti $q - 1$ nista deljiva s 3. V tem primeru je šifriranje zelo hitro, saj porabi le eno modularno množenje in eno modularno kvadriranje. Drug uporabljen šifrirni eksponent v praksi je $e = 2^{16} + 1 = 65537$. To število ima le dve 1 v binarni reprezentaciji, tako da šifriranje z algoritmom kvadriraj-in-množi izvede le 16 modularnih kvadriranj in eno modularno množenje. Šifrirni eksponent $e = 2^{16} + 1$ je boljša izbira od $e = 3$, saj je odporen na napade, obravnavane v §4.3.

6.2 RSA elektronski podpis v praksi

Z uveljavitvijo elektronskega poslovanja prihaja do vedno večje potrebe po elektronskem podpisu. RSA kriptosistem je zelo primeren za implementacijo elektronskega podpisa. Poleg RSA elektronskega podpisa (§3.4), ki je shema elektronskega podpisa s sporočilom, obstajajo različne sheme elektronskega podpisa, ki temeljijo na RSA kriptosistemu. V §5.9 smo opisali shemo IFSSA, shemo elektronskega podpisa z dodatkom, ki je del pred kratkim nastalega standarda IEEE P1363 [14]. Pri RSA elektronskem podpisu lahko omenimo naslednje probleme, s katerimi se srečujemo v praksi.

(i) Problem blokiranja

Eden od priporočenih načinov uporabe RSA elektronskega podpisa je ta, da sporočilo najprej podpišemo, potem pa ga še zašifriramo. Vendar moramo upoštevati relativno velikost obeh modulov, ki ju pri tem uporabimo.

Recimo, da želi oseba *A* podpisat, kasneje pa še zašifrirat sporočilo za osebo *B*. Predpostavimo, da sta (n_A, e_A) in (n_B, e_B) javna ključa za *A* in *B*. Oseba *A* izračuna

$$s = m^{d_A} \bmod n_A \text{ in } c = s^{e_B} \bmod n_B,$$

oseba *B* pa preveri podpis in pridobi sporočilo tako, da izračuna

$$s' = c^{d_B} \bmod n_A \text{ in } m' = s'^{e_A} \bmod n_B.$$

Če je $n_A > n_B$, potem obstaja možnost, da oseba *B* sporočila ne more pridobiti iz podpisa, tj. $m' \neq m$. Razlog je v tem, da je *s* večji od n_B. Verjetnost, da se to zgodi je $(n_A - n_B)/n_A$. Imamo več načinov, da se izognemo temu problemu:

1. *drugačen vrstni red*. Napačno dešifriranje se ne bo zgodilo, če naredimo najprej operacijo z manjšim modulom. Če je $n_A > n_B$, potem naj oseba *A* najprej zašifrira sporočilo z javnim ključem osebe *B*, potem pa ga podpiše s svojim zasebnim ključem. Vseeno je zaželjen vrstni red operacij podpis sporočila, potem pa šifriranje, kajti če oseba *A* najprej zašifrira in potem podpiše, lahko nasprotnik odstrani podpis in ga zamenja s svojim. Tudi če nasprotnik ne ve, kaj je podpisal, je to v določenih primerih lahko dobro zanj. Zato drug vrstni red ni vedno najboljša rešitev.

2. *dva modula ene osebe*. Če imajo vse osebe dva modula (enega za šifriranje, drugega za podpisovanje) in so vsi podpisni moduli manjši od vseh šifrirnih modulov, potem se napačno dešifriranje nikoli ne zgodi. To lahko naredimo tako, da predpišemo dolžino šifrirnih modulov na $(t + 1)$-bitov, podpisnih modulov pa na dolžine *t*-bitov, za neko število *t*.

3. *predpisane lastnosti modulov*. S to metodo izberemo taki praštevili *p* in *q*, da ima modul *n* posebno obliko. Npr. bit z najvišjim redom je enak 1, naslednjih nekaj bitov pa 0. Ta izbira za modul *n* sicer ne prepreči popolnoma napačnega dešifriranja, vendar pa zmanjša verjetnost tega dogodka.

(ii) Učinkovitost izdelave in preverjanja podpisa

Naj bo $n = pq$ modul dolžine $2k$-bitov, kjer sta *p* in *q* oba dolžine *k*. Zahtevnost izračuna podpisa $s = m^d \bmod n$ na sporočilu *m* je $O(k^3)$ (§5.4 (iv)). Ker oseba, ki naredi podpis, ponavadi pozna praštevili *p* in *q*, lahko izračuna $s_1 = m^d \bmod p$ in $s_2 = m^d \bmod q$ in določi podpis *s* s Kitajskim izrekom o ostankih (§5.5). Čeravno zahtevost tega postopka ostane $O(k^3)$, je le-ta v nekaterih primerih občutno učinkovitejši. Preverjanje podpisa je občutno hitrejše od podpisovanja, če je javni eksponent majhno število. V tem primeru je ocena zahtevnosti $O(k^2)$. Tako kot pri šifriranju so tudi tu predlagane vrednosti za šifrirni eksponent *e* v praksi enake 3 ali $2^{16} + 1$.

(iii) Izbor parametrov

Od leta 1996 je minimalna dolžina RSA podpisnega modula 768 bitov. Modul, dolžine vsaj 1024 bitov, pa je priporočen za podpise, ki zahtevajo daljši življenski čas ali so kritični za celotno varnost velikega omrežja. Preudarno je spremljati napredek pri faktorizaciji velikih števil in biti pripravljen izbrati primerne parametre.

Do zdaj ni bilo opažene nobene slabosti pri RSA elektronskem podpisu, ki se tiče izbire 'majhnega' javnega eksponenta *e*, kot sta 3 ali $2^{16} + 1$. Ni pa priporočljiva omejitev velikosti zasebnega eksponenta *d* za to, da bi se povečala učinkovitost generiranja podpisa (glej §4.3).

(iv) Kratka in dolga sporočila

Recimo, da imamo $2k$-bitni RSA modul n za podpisovanje k-bitnih sporočil in želimo podpisati $(k \cdot t)$-bitno sporočilo m. En pristop je, da razdelimo m na k bitne bloke, $m = m_1 \| m_2 \| \ldots \| m_t$, in vsak blok podpišemo posebej, vendar to ni priporočljivo, saj je generiranje podpisov relativno počasno. Alternativno lahko zgostimo sporočilo m v niz dolžine $l \leq k$ in podpišemo zgoščeno vrednost. To pomeni, da moramo uporabiti elektronski podpis z dodatkom. Za sporočilo dolžine največ k bitov, je boljše uporabiti elektronski podpis s sporočilom.

6.3. Patenti in standardi

Kriptografski patenti po eni strani javnosti predstavljajo pomembne dosežke na področju kriptografskih procesov in učinkovite načine uporabe le-teh, po drugi strani pa omejujejo njihovo uporabo zaradi potrebnih licenc. Patenti imajo v ZDA veljavno dobo 17 let od datuma izdaje ali 20 let od datuma vložitve.

RSA kriptosistem je patentiran v ZDA in Kanadi. Patent Rivest-Shamir-Adelman je bil vložen 14. 12. 1977. Datum izdaje patenta RSA je bil 20. 9. 1983. Patent je bil 6. 9. 2000 izpuščen v javnost. S tem je postal RSA kriptosistem zelo zanimiv, saj za njegovo uporabo ni več potrebna licenca. Patent RSA pokriva RSA šifrirno shemo (§3.3) in RSA elektronski podpis (§3.4). Omenjene so tudi posplošitve, kot so npr. uporaba modula n, ki je produkt treh ali več praštevil, podpis stisnjenega sporočila in uporaba RSA šifriranja za prenos ključev.

Hellman-Bachov patent pokriva metodo za generiranje RSA ključev, torej praštevil p in q in modula $n = pq$, ki zadoščajo nekaterim lastnostim, za katere se verjame, da je RSA nezlomljiv. Patent je bil vložen 31. maja 1984. Vključuje standardne pogoje za močna praštevila (glej §6.1 (ii)).

Kriptografski standardi olajšajo in s tem pospešujejo uporabo konkretnih splošno sprejetih kriptografskih rešitev in zagovarjajo uporabo le-teh pri implementaciji v varnostne mehanizme in ostale sisteme. Izdelanih je (ali pa so še v izdelavi) več standardov, ki se nanašajo na uporabo RSA kriptosistema za šifriranje, digitalne podpise in vzdrževanje ključev.

Standard ISO/IEC 9796 specificira mehanizme za sheme elektronskega podpisa s sporočilom. Vsebuje tudi primer za RSA šifrirno shemo. Glavni del standarda je del o funkciji R (glej §3.4), ki je splošno uporabna za razne sheme, med drugim tudi RSA shemo, ker izključuje napad §4.4 (ii).

Standard ISO/IEC 14888 govori o elektronskem podpisu z dodatkom. Vsebuje pregled potrebnih korakov pri procesu generiranja podpisov in različne načine preverjanja podpisov za različne kriptosisteme, tudi za RSA kriptosistem.

IEEE P1363 je postal standard v začetku leta 2000. Nastal je pri Institute of Electrical and Electronics Engineers [14]. Vsebuje standardne specifikacije za javno kriptografijo - šifrirne sheme, sheme za prenos ključev in elektronske podpise, ki temeljijo na RSA in drugih javnih kriptosistemih, med drugim tudi IFSSA (§5.9), shemo elektronskega podpisa z dodatkom, ki temelji na RSA kriptosistemu in zgoščevalni funkciji SHA-1 (§5.8).

7. VIRI

[1] BACH, E., SHALLIT, J.: *Alghorithmic Number Theory, Volume I: Efficient Algorithms*, Mit Press, 1996.

[2] BONEH, D.: *Twenty Years of Attacks on the RSA Cryptosystem*, Notices of the American Mathematical Society, **46** (2), 1999, 203-213.

[3] BONEH, D., DURFEE G., FRANKEL, Y.: *An attack on RSA given a fraction of the private key bits*, ASIA-CRYPT '98, Lecture Notes in Computer Science, 1514, Springer-Verlag, 1998, 25-34.

[4] BONEH, D., DEMILLO, R., LIPTON, R.: *On the importance of checking cryptographic protocols for faults*, EUROCRYPT '97, Lecture Notes in Computer Science, 1233, Springer-Verlag, 1997, 37-51.
(glej http://ftp.cryptography.com/resources/papers/index.html)

[5] BONEH, D., VENKATESAN, R.: *Breaking RSA may not be equivalent to factoring*, EUROCRYPT '98, Lecture Notes in Computer Science, 1403, Springer-Verlag, 1998, 59-71.

[6] CHILDS, L.: *A Concrete Introduction to Higher Algebra*, Springer-Verlag, 1977

[7] COHEN, H.: *A Course in Computational Algebraic Theory*, Springer-Verlag, 1993.

[8] CONTINI, S.: *The factorization of RSA-140*, RSA Laboratories' Bulletin, **10**, 1999.
(glej http://www.rsasecurity.com/rsalabs/bulletins/)

[9] COPPERSMITH, D.: *Small solutions to polynomial equations and low exponent RSA vulnerabilities*, Journal of Cryptology, **10**, 1997, 233-260.

[10] COPPERSMITH, D., FRANKLIN, M., PATARIN, J., REITER, M.: *Low-exponent RSA with related messages*, EUROCRYPT '96, Lecture Notes in Computer Science, 1070, Springer-Verlag, 1996, 1-9.
(glej http://ftp.cryptography.com/resources/papers/index.html)

[11] DIFFIE, W., HELLMAN, M.E.: *New directions in cryptography*, IEEE Transactions on Information Theory, **22**, 1976, 644-654.

[12] HARDY, G.H., WRIGHT, E.M.: *An introduction to the Theory of Numbers*, Oxford Univ. Press, 1962.

[13] HASTAD, J.: *Solving stimultaneous modular equations of low degree*, SIAM Journal of Computing, **17**, 1988, 336-341.

[14] IEEE Standart Department: IEEE P1363 / D13 (Draft Version 13), *Standard Specifications for Public Key Cryptography*, 1999.
(glej http://grouper.ieee.org/groups/1363/index.html)

[15] JURIŠIĆ, A., TROJAR, A.: *Pametna kartica*, Uporabna informatika, **5**, 1997, 37-45.

[16] KAHN, D.: *The Codebreakers*, Macmillan Publishing Company, 1967.

[17] KOBLITZ, N.: *A Course in Number Theory and Cryptography*, Springer-Verlag, 1987.

[18] KOCHER, P. C.: *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems*, CRYPTO '96, Lecture Notes in Computer Sciences, 1109, Springer-Verlag, 1996, 104-113.
(glej http://ftp.cryptography.com/resources/papers/index.html)

[19] MENEZES, A., OORSCHOT, P., VANSTONE, S.: *Handbook of Applied Cryptography*, CRC Press, 1997.
(glej http://www.cacr.math.uwaterloo.ca/hac)

[20] PAPADIMITRION, C. H., STEIGLITZ, K.: *Combinatorial Optimization Algorithms and Complexity*, Prentice Hall Inc., 1982.

[21] RIVEST, R. L., SHAMIR, A., ADLEMAN, L.: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, **21** (2), 1978, 120-126.

[22] SHAND, M., VUILLEMIN, J.: *Fast Implementations of RSA Cryptography*, Proceedings of the 11th IEEE Symposium on Computer arithmetic, 252-259, 1993.

[23] STINSON, D. R.: *Cryptography - Theory and Practice*, CRC Press, 1995.

[24] VIDAV, I.: *Algebra*, Formatisk, 1989.

[25] WIENER, M.: *Cryptoanalysis of short RSA secret exponents*, IEEE Transactions on Information Theory 36, 1990, 553-558.

INDEKS